Master's thesis

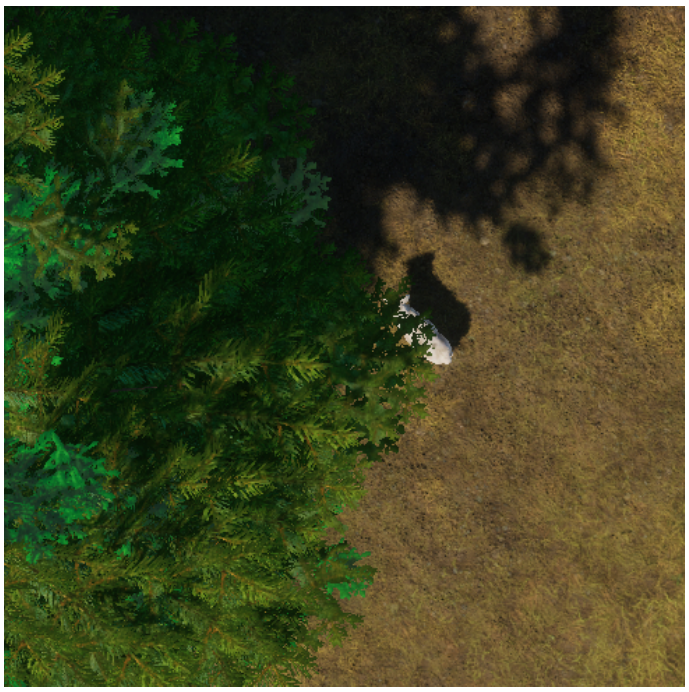
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jørgen Borander
Sindre Langaard

Leveraging Synthetic Data for Boosting Occluded Sheep Detection in UAV Images

Master's thesis in Computer Science, Informatics
Supervisor: Svein-Olaf Hvasshovd

June 2023



NTNU
Norwegian University of
Science and Technology

Jørgen Borander
Sindre Langaard

Leveraging Synthetic Data for Boosting Occluded Sheep Detection in UAV Images

Master's thesis in Computer Science, Informatics
Supervisor: Svein-Olaf Hvasshovd
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Each fall, Norwegian farmers round up around 2 million sheep and lambs from grazing. Locating and gathering all the sheep can be time-consuming and tedious, considering some stray sheep may have wandered far beyond their designated grazing areas. With technological advancements, new tools have emerged to assist with sheep roundups, such as using unmanned aerial vehicles UAV to locate sheep. Previous research has examined applying machine learning algorithms to images captured by UAVs to detect sheep automatically, showing promising results and capabilities. However, not all sheep are detected as easily. Sheep which are partially occluded by vegetation has proved especially difficult to detect.

The aim of this thesis is to improve occluded sheep detection accuracy by training object detection models on synthetic and real image data. An approach to generating synthetic images of occluded sheep data in Unity Perception is explored. The generated synthetic data and real image data are used to train a range of YOLOv7 models. A baseline model trained on real data exclusively, a model trained on mixed data, and a fine-tuned model are examined to evaluate its impact on occluded sheep detection performance. The performance from mixed training and fine-tuning will also be compared.

The results indicate that synthetic image data can boost the performance of occluded sheep detection. Both the mixed and fine-tuned models trained with synthetic data performed better than the baseline model, with the mixed-trained model seeing the most significant boost in performance.

Sammendrag

Hvert høst samler norske bønder inn omtrent 2 millioner sauer og lam fra beite. Å lokalisere og samle alle sauene kan være en tidkrevende og langtrukken prosess da noen av sauene kan ha vandret langt utenfor sitt opprinnelige beiteområde. Med teknologiske fremskritt har det dukket opp nye verktøy for å hjelpe med sauesanking, som bruken av droner for å finne sau. Tidligere studier har sett på å anvende maskinlæringsalgoritmer på bildene tatt av droner for detektere sau automatisk, og det har vist lovende resultater og muligheter. All sau er derimot ikke like enkel å oppdage. Å detektere sau som er gjemt under vegetasjon har vist seg å være spesielt utfordrende.

Målet med denne oppgaven er å øke nøyaktigheten av deteksjon av skjult sau ved å trene objektdetekteringsmodeller på syntetiske og ekte bilde-data. En tilnærming til å generere syntetiske bilder av sauer delvis skjult av vegetasjon i Unity Perception blir utforsket. De genererte syntetiske bildene og ekte bilde-data blir brukt til å trene en rekke YOLOv7-modeller. En referansemodell trent utelukkende på virkelige data, en modell trent på en kombinasjon av syntetisk og ekte data, og en finjustert modell blir undersøkt for å vurdere effekten på ytelsen ved deteksjon av sauer delvis skjult av vegetasjon. Hvordan trening på kombinasjonen av syntetisk og ekte data sammenlikner med en fin-justert modell vil også bli utforsket.

Resultatene indikerer at syntetiske bilde-data kan forbedre ytelsen ved deteksjon av sauer delvis skjult av vegetasjon. Både modellen trent på kombinerte data og den finjustert modellen presterer bedre enn referansemodellen, hvor modellen trent på kombinerte data oppnådde den største forbedringen i ytelse.

Acknowledgements

This master's thesis is a part of the MIDT and MSIT master programmes at the Department of Computer Science at the Norwegian University of Science and Technology.

We want to thank our supervisor Svein-Olaf Hvasshovd, for his guidance and encouragement while writing this thesis. No meeting without a smile, laughter and encouraging words. Thank you to Bjørnar Østtveit, Hallvard Stemshaug and Kari Meling Johannessen for sharing their data and insights from previous experiments and work.

A special thanks to all the other master's students at Gamle Fysikk, who have motivated us throughout the process. Their support and the weekly *vinlotteri* and *kliss og quiz* have made working on the thesis a joy.

Contents

Figures

Tables

Acronyms

ANN	Artificial neural network. 20, 22
AP	Average precision. 7, 19, 20, 54
CNN	Convolutional neural network. 9, 22, 23, 25, 44
DIMO	Dataset of industrial metal objects. 7
E-ELAN	Extended efficient layer aggregation network. 23
FN	False negative. 14, 17
FP	False positive. 14, 17
GANs	Generative adversarial networks. 8, 9, 43, 70
GPS	Global positioning system. 38
IoU	Intersection over union. 14, 16, 20
IR	Infrared radiaton. 6, 38
mAP	Mean average precision. 8, 9, 20, 56
MS COCO	Microsofts common objects in context. 7, 9, 27, 47
NIBIO	Norsk institutt for bioøkonomi. 13
NMS	Non maximum suppresion. 16
NTNU	Norwegian University of Science and Technology. 5, 6, 8, 37
RGB	Red green blue. 6
SOTA	State of the art. 2, 63
SSD	Single-shot detector. 44

TN True negative. 14

TP True positive. 14, 17

UAV Unmanned aerial vehicle. iii, 1–3, 5, 6, 30

YOLO You only look once. 2, 5, 6, 8, 23, 25, 27, 44

Chapter 1

Introduction

1.1 Background and Motivation

Each year around 2 million sheep and lamb are released into Norwegian nature for grazing [1]. At the end of the grazing period in the fall, the sheep farmer rounds up the sheep for the winter. Sheep roundup can be a challenging and tedious task, as the sheep move over a large area and can be spread across difficult-to-predict locations and a range of terrain [2]. Through grazing trips, the farmer tries to round up all the sheep, but because of the large area to cover, not all sheep are likely to be found [3]. Farmers are often required to complete several grazing trips to locate all of their sheep. Finding the last ones can require a large crew searching over time, quickly becoming expensive [2]. Failing to locate the last sheep can lead to a loss in revenue, and the resources used in the search could have been better spent elsewhere.

Because of this, the farmer uses tools to help with sheep roundup. Traditionally, bells are attached to the neck of the sheep to help with localisation. This is helpful as the bell makes a lot of noise when the sheep are moving about, but it requires the farmer to be relatively close to hear the sound. Neither do the bells create sound when the sheep are stationary, so they will not help locate dead or heavily injured sheep. There are, however, newer technologies made for helping with sheep roundups [4][5][6][7]. These devices are tracking sheep using GPS and transmitting location data to the farmer over satellite or cellular connectivity. Being battery driven, they can be recharged and used for years, some even having a life expectancy spanning multiple seasons on one charge. They can, however, have a significant upfront cost, and most require an active subscription. These alternatives are helpful but can be expensive for the farmer.

Camera-equipped Unmanned aerial vehicles (UAVs) have also proven to be an effective way of rounding up sheep [8][9][10][11]. It is more affordable and can be effective at scanning through the terrain. These UAVs fly over the landscape, capturing images looking for sheep, which a machine learning algorithm processes to detect sheep automatically. This method has proven to be an effective way of finding sheep[12]. However, it has been shown that some sheep are more challenging to detect than others, primarily sheep with darker wool or sheep occluded by vegetation [13]. These sheep have a limited amount of image data associated with them, making training an object detection algorithm a challenge [14]. Ways of creating synthetic data have been proposed and show that it can positively affect the training of models and detection of sheep in the wild [13][15], however, detecting occluded objects remain to be a challenge [16]. This thesis will build upon the work on generating synthetic data to train object detection algorithms, focusing on improving accuracy when looking at occluded sheep.

1.2 Thesis Goal and Research Questions

Goal: *Improve detection of partially occluded sheep by training YOLOv7 on real and synthetic data generated with Unity Perception*

The main goal of this thesis is to improve the detection performance of partially occluded sheep by training the You only look once (YOLO) version 7 algorithm on real and synthetic data generated with Unity Perception. Using synthetic data to train an object detection model, the hypothesis is that it can improve the detection accuracy of occluded sheep.

The goal is further broken down into three research questions addressing key factors regarding using synthetic data and detecting occluded sheep. Answering these questions will help with reaching the overall goal for the thesis.

RQ1: *How does YOLOv7 perform with the dataset consisting of real images?*

Previous work looking at the detection of sheep in UAV images done by *Østtveit* [13] used the YOLO version 5 architecture. Since then, the object detection field has progressed, with improved algorithms being released. Comparing how a State of the art (SOTA) object detection algorithm performs on the existing dataset will give an updated baseline to compare the results from adding occluded images.

RQ2: *How does training on mixed synthetic and real data affect the detection performance of occluded sheep?*

The hypothesis is that generating more image data with hard-to-detect classes and using this for training will yield better results. These hard-to-detect classes are sheep which are partially occluded by vegetation. Rendering images of sheep occluded by trees, branches and bushes and using these for training should improve the accuracy performance of the model.

RQ3: *How does fine-tuning compare to mixed training performance for predicting occluded sheep?*

Various techniques exist for training object detection models using synthetic data. Examining the performance differences between two methods, mixed training and fine-tuning, can give insight into how object detection models for sheep detection should be trained.

1.3 Scope and Limitations

This thesis focuses on improving the performance of detecting occluded sheep in UAV images. The rest of the system for sheep detection, such as the UAV itself, controls, running the object detection model, or designing the graphical interface handled by the user, is outside the scope of the thesis.

A limiting factor has been the image data available for training. Supplying the dataset with more images has not been possible as the UAV has been missing for several months, only surfacing in the spring. As the writing of this thesis doesn't match the season where the UAV is intended to be used, it wouldn't be optimal to capture new images in the spring.

Time has also been a limiting factor for training and generating images. Training a model takes many hours, and suitable hardware is in high demand and not always available, often resulting in days of waiting in queues. Generating synthetic data also requires time. Therefore, testing every possible configuration and possibility for improvement of the object detection models has not been feasible.

1.4 Thesis Structure

Chapter 1: Introduction gives an overview of the problem description and presents the goal, research questions and scope for the thesis.

Chapter 2: Related Work presents related work and research done on relevant subjects within sheep roundup and object detection.

Chapter 3: Theory and Background Knowledge gives an introduction to background knowledge on the domain and introduces relevant theory from the field of computer vision and object detection.

Chapter 4: Method explains the method of generating occluded sheep data, presents existing datasets and explores alternative approaches for generating synthetic data and training the object detection algorithm.

Chapter 5: Experiment Structure presents how the experiments for evaluating the research questions are structured.

Chapter 6: Results presents the results from training and testing following the data generation and training.

Chapter 7: Discussion examines the results from testing and discusses their meaning, validity and implications with regard to the research questions.

Chapter 8: Conclusion evaluates the goal of the thesis, its contribution, and suggestions for future work.

Chapter 2

Related Work

The following chapter will present related work and research on subjects relevant to this thesis. It will cover some of the previous master thesis about using Unmanned aerial vehicle (UAV)s for sheep roundup and research on different aspects of object detection, including using synthetic data and the problems of class imbalance and occlusion.

2.1 Master Thesis on Sheep Detection in UAV Images

As UAVs have become viable for agricultural tasks such as monitoring crops and livestock [12], studies and theses have looked at the application and usability of UAVs for sheep roundup. The following theses are written by students from the Norwegian University of Science and Technology (NTNU) under the supervision of Professor Svein-Olav Hvasshovd, looking at different aspects of sheep detection in UAV images.

Muribø [10] experimented on how well the object detection algorithm YOLO version 3 performed when detecting sheep. He examined performance when detecting sheep as a superclass versus as three subclasses: white, black and brown. He also looked at how tuning input resolution and confidence threshold when training affected performance. The results showed that detecting sheep as a superclass with a resolution of 832x832 pixels and a confidence threshold of 0.1 performed the best. With these calibrations, the algorithm detected 1638 of 1650 sheep, giving it a recall score of 0.99. However, there are doubts as to how applicable Muribøs results are. The dataset for training the algorithm consisted primarily of white sheep, without much variation in landscape and light conditions. He theorises that with more and varied data, the findings may apply to real-use scenarios.

Furseth and Granås [17] studied the use of YOLOv5-based deep learning models, testing configurations of different sizes and training on different image types and resolutions. These models were evaluated on their accuracy and inference speed running on mobile devices with limited hardware. Comparing YOLOv5s and YOLOv5m, they found that the larger YOLOv5m with more trainable parameters performed only marginally better than the smaller YOLOv5s, however, the difference was more significant at lower resolutions. Inference time was significantly higher for YOLOv5m running on a mobile device. They also found that training on tiled high-resolution images performed better than training on the same downscaled images.

Johannessen [8] proposed a deep learning model which could detect sheep in RGB and IR images captured from a UAV. The proposed model fused Red green blue (RGB) and Infrared radiaton (IR) images to utilise the features from both formats. She compared the performance of the fusion model to strictly RGB and IR models' results. The results show that the most accurate models with a confident threshold of 0.5 achieved a grid precision of 0.977 and a recall of 0.901, corresponding to the detection of 97.5 % of the sheep in the validation dataset. Her results suggest that RGB and IR fusion images can effectively detect sheep.

However, Stemshaug [18] found contrary results. He compared the performance of YOLOv5 models fusing IR and RGB with models using only RGB images to detect sheep. He experimented with different configurations to determine the impact of the IR images and whether they increased performance enough to justify the increased price of a UAV equipped with both cameras. His results suggest that the highest-performing model uses tiled RGB images to locate four classes of sheep, having a recall of 0.986. The highest-performing model fusing RGB and IR images had a recall of 0.960. Even though these results are comparable, his research findings suggest that the actual value of IR images doesn't materialise, as the UAV isn't used in colder darker environments, so a UAV without IR image capabilities is sufficient for the use case of sheep roundup.

2.2 Locating Sheep with Radio-Equipped UAVs

Alternative approaches for using UAVs for sheep roundups are also possible. Using radio transmitters instead of cameras has been explored by NTNU students to create systems for tracking sheep.

Vucic and Axell [19] studied using a system consisting of a UAV, lightweight radio tags on sheep, and a custom Ground Control Station to locate sheep. The system used Bluetooth Low Energy, Round Trip Time, and trilateration to find sheep. Using cheap radio tags, tracking a large percentage of the herd is economically feasible. However, the system is not thoroughly tested to the extent it needs to be a reliable solution for the farmer, as different terrain can affect the system's performance. They found it successful in locating sheep in open fields and light forests but struggled in heavier forestation. More research is required to determine if the system is viable for real-use scenarios.

2.3 Object Detection with Synthetic Data

As object detection has spread to more use cases, the need for quality data has increased. Capturing images of real-world examples might not always be feasible, and if it can be captured, manually labelling data is tedious and is prone to human error [20]. Because of these disadvantages, training object detection algorithms with synthetic data has been explored as a possibility.

An analysis by Vanherle *et al.* [15] from 2022 examined training object detection models with synthetic training data. A problem addressed in the paper is the resemblance between real and synthetic data. The synthetic data might look realistic, but there is still a difference between the real and rendered data. This is called the *domain gap* [21], and because of this, models trained on synthetic data will perform worse than when trained on real data. The author looks at different methods and techniques during training to overcome this problem. Using the Dataset of industrial metal objects (DIMO) [22] dataset allowed for controlled variations and randomisation in the synthetic dataset. When generating data, they compared the effects of using real data, synthetic copies, synthetic data with random poses, synthetic data with random lighting, and synthetic data with random poses and lighting. Their results suggest that mimicking high-level features, such as poses and shapes, is easier to make look realistic than low-level features, such as textures, lights, and colours. Their findings indicate that it is beneficial to accurately simulate high-level features while randomising low-level features, as this made the model generalise best in the experiment.

Different techniques for training with synthetic data were also analysed. When comparing transfer learning and data augmentation for better performance, they found that transfer learning performed the best, with quicker convergence speed and better Average precision (AP) scores. With transfer learning from the Microsofts common objects in context (MS COCO) dataset, the AP scores were high from the start, meaning even a few thousand images can produce a decent model. Leveraging real images boosted the models' performance even more, with fine-tuning outperforming mixed training for all real and synthetic data ratios. Adding more synthetic images increased mixed training performance, maximising at a ratio of 5 synthetic images for each real. The best-performing fine-tuned model had a ratio of 1:1. They concluded that using both real and synthetic images improved performance, with fine-tuning being the recommended approach.

Seib *et al.* [23] looked at current approaches for using synthetic data in neural network training. As Vanherle *et al.* [15], they looked at transfer learning and data augmentation with some variations. Pre-training a model before fine-tuning was looked at but with different results. They found that pretraining a model and then freezing all layers without an application-specific purpose was desirable, in contrast to Vanherle *et al.*, who found that even a pre-trained model could benefit from continuing to tweak the shallow layers of the network. Seib *et al.* also looked at other ways of data augmentation for training. In addition to simple transformations like flipping, cropping, or otherwise distorting images to produce more variations, they also looked at random erasing, sample pairing and using Generative adversarial networks (GANs) to extend the dataset. Random erasing, where an arbitrary block of pixels is substituted with black pixels, and sample pairing, where two classes are averaged along each RGB channel to create new examples during training, have reported improved model performance. GANs have the potential to create new example images but require a lot of data to be able to generate new examples.

Borkman *et al.* [24] introduced the *Unity Perception* package in 2021. It is a tool for generating labelled synthetic data for computer vision tasks. It generates custom 3D scenes with randomisation and variation and automatically creates labels and annotations, considerably speeding up the process of generating synthetic data. They used the Faster R-CNN model with the ResNet50 backbone pre-trained on the ImageNet dataset for testing. They compared the performance of the model trained on a real image dataset versus the synthetic dataset fine-tuned on the real data. The real image dataset consisted of 1267 images with 63 classes of groceries, and the synthetic dataset consisted of 400,000 synthetic images. The baseline model, trained on the real dataset, achieved a Mean average precision (mAP) of 0.719, while the model trained on only synthetic data achieved a mAP of 0.538. However, by fine-tuning the model trained on synthetic data with the full real dataset, they achieved the highest mAP of 0.854. Their research suggests that Unity Perception is a viable option for synthetic image generation and can potentially boost detection performance.

Another NTNU student, Østtveit [13], used synthetic data generated with Unity Perception to detect sheep in UAV images. He examined which categories of sheep an object detection algorithm finds the most difficult to detect, being the dark-wooled and occluded classes. He then trained the YOLOv5 algorithm on synthetic and real images to boost the detection of these difficult-to-detect classes, achieving higher recall scores. The results indicate that training a model on a combination of real and synthetic sheep images is viable for sheep detection when UAV image data is limited.

2.4 Imbalanced Classes in Object Detection

When the distribution of classes in a dataset is not uniform, it can lead to biased models favouring the majority classes [25]. This is a relevant issue in object detection, and it is essential to have a varied dataset to ensure that the algorithm learns features from all classes.

A review by Oksuz *et al.* [14] from 2020 looked at the state of imbalance problems in machine learning. Looking at deep-learning object detection literature, they identified eight different imbalance problems, grouped into four types; class imbalance, scale imbalance, spatial imbalance, and objective imbalance. The most relevant category for this thesis is class imbalance, in which one class is over-represented with more examples than others. It can occur in two different ways. In foreground-background class imbalance, the over- and under-represented classes are foreground and background classes, respectively, meaning there are more negative examples than positive ones. With object detectors, this is inevitable, as most bounding boxes will be labelled as negative. In foreground-foreground imbalance, there is an imbalance between the classes in the dataset. An approach to this problem is using generative methods to address the imbalance and supply underrepresented classes with more examples. Methods utilising GANs are suggested, but using other data generation methods could also be viable, like using a game engine.

2.5 Occlusion in Object Detection

Occlusion is when an object is hidden or partially hidden by another object in an image. It can occur by the same type of object called intra-class occlusion or another object or a fixed element called inter-class occlusion [16]. Occlusion can occur in many ways, as objects of different sizes and degrees of coverage can occlude the object, making it challenging to tackle on a general basis [16].

Wang *et al.* [26] examined data creation for different ways occlusion can occur. They suggest utilising adversarial networks to generate synthetic images of hard examples with occluded and deformed objects. These hard examples are used to train the object detection model and adversarial network simultaneously. They reported increases in mAP score of 2.3% on the VOC07 dataset and 2.6% on the VOC2012 dataset compared to the Fast-RCNN, showing the potential of GANs and generated synthetic images of occluded objects.

Kortylewski *et al.* [27] introduced Compositional Convolutional Neural Network (CompositionalNet). They unified Convolutional neural network (CNN) with compositional models by replacing the fully-connected classification head of a CNN with a differentiable generative compositional model. After experimenting on real and synthetic occluded images from the MS COCO and PASCAL3D+ dataset, they found that the CompositionalNet model outperformed standard CNNs by a large margin for classifying and detecting vehicles under occlusion, showing the potential of generative models for boosting detection of occluded objects.

Chapter 3

Theory and Background Knowledge

The following chapter will introduce relevant background theory for the thesis. It will cover sheep and sheep roundup knowledge, object detection and relevant metrics, neural networks, insight into YOLOv7, and aspects of training object detection models.

3.1 Sheep Roundup

Approximately 2 million sheep and lamb are on the pasture during the grazing season [1]. During the season, the sheep farmer must do weekly welfare checks on the sheep. This is required by law, so the farmer must keep track of the sheep's whereabouts during the season. Both end-of-the-season roundup and welfare checks can be challenging as the sheep roam relatively freely through the terrain. Fences and salt blocks are placed throughout the fields to keep the sheep from roaming too far away, but some might still be far away from their designated grazing area.

Bringing the sheep back from the pasture is primarily done in three steps [2]:

Primary roundup:	For 1-2 weekends, a larger group of people and shepherd dogs help the farmer round up the sheep. In this phase, approximately 90% of the sheep are found.
Secondary roundup:	With less help, the farmer searches for missed sheep in the same areas as in the primary roundup. This phase lasts a few weeks, and most remaining sheep are found.
Find stray sheep:	In the last phase, the farmer searches beyond the primary pasture to find the last sheep. They might have travelled far from the primary pasture, so finding the last ones can be challenging.

The effort and time required during the final phase of roundup can often be a source of frustration for the farmer. This is where the need for effective tools to help with sheep roundups emerges.

3.1.1 Existing Tools for Sheep Tracking

Bells

Traditional bells have been used for keeping track of sheep for hundreds of years. It gives the farmer an indication of where the sheep are, as the bells create a lot of noise when the sheep are moving. The bells are attached to the neck of the sheep, making it a cheap and practical tool for locating sheep.

However, bells also have their downsides. The sound of the bell is loud, but the farmer still needs to be in relative proximity to be able to hear it. Secondly, the constant sound of the bell can damage the sheep, as there needs to be more research into how its hearing and well-being are affected [28]. The sound of the bell also might attract predators who can hear the sound at further ranges than humans [28].

Radio bells

Over the last 20 years, *radio bells* have been introduced as a viable option to traditional bells. Different solutions are available on the market, such as models from Telespor [6], Findmy [5], and Nofence [29]. These options vary in price and features. **Table 3.1** compares different models available on the market.

	Telespor Radiobjella	*FindMy Modell 2 Rød*	*Nofence Småfeklave*
Price (per unit)	1,088 kr	2,490 kr	1,950 kr
Subscription (per unit)	144 kr	169 kr	334 kr
Network technology	GNSS, LTE-M, Bluetooth	LTE, GNSS	LTE-M, GNSS, Bluetooth
Battery life (seasons)	1	2-3	1
GPS tracking	x	x	x
Geofencing	-	x	x
Movement alarm	x	-	-
Stress Warning	-	x	-
Waterproof	x	-	-

Table 3.1: Comparisons of tracking systems from Telespor, FindMy and Nofence

These solutions have features that allow the farmer to control where the sheep can move with geofencing and real-time tracking of their whereabouts. Some even have stress warning alarms, notifying the farmer if there is a disturbance in the sheep herd, requiring additional inspection.

However, connectivity can be a problem for these tracking systems, as they rely on different technologies with varying degrees of coverage in the pasture, like LTE.

Cellular providers typically do not cover grazing pastures, decreasing their effectiveness or proving them worthless. Some newer models have developed solutions and workarounds for the lack of connectivity, using base stations instead of cellular coverage [4]. While cellular coverage is steadily increasing, these challenges persist and continue to pose difficulties for some farmers.

Another issue with radio bells is their associated cost. Various providers suggest different levels of coverage of tracked sheep. Nofence recommends fitting all adult sheep with a tracker [29], while FindMy recommends a minimum coverage of 25% [5]. As stated by Johanssen and Sørheim [30], it is important to consider that a sheep herd typically resides in smaller groups of 8-10 sheep. Because of this, not all sheep need their own radio bell, making it possible to tag a smaller percentage of the herd. **Table 3.2** gives an overview of the cost of different commercial solutions at varying degrees of coverage. It is based on Norsk institutt for bioøkonomi (NIBIO)s numbers from 2021, which state that the average sheep farmer has 142 winter-fed sheep [31].

Coverage	*Costs*	*Telespor Radiobjella*	*FindMy Modell 2 Rød*	*Nofence Småfeklave*
25%	Up-front	38,624 kr	88,395 kr	69,225 kr
	Seasonal	5,112 kr	6,000 kr	11,857 kr
50%	Up-front	77,248 kr	176,790 kr	138,450 kr
	Seasonal	10,224 kr	11,999 kr	23,714 kr
75%	Up-front	115,872 kr	265,185 kr	207,675 kr
	Seasonal	15,336 kr	17,999 kr	35,571 kr
100%	Up-front	154,496 kr	353,580 kr	276,900 kr
	Seasonal	20,448 kr	23,998 kr	47,428 kr

Table 3.2: Comparing cost of Telespor, FindMy and Nofence systems at various tracking levels

These figures show the substantial cost associated with tracking sheep. While these estimates are not entirely precise, they indicate the potential expenditure tied to the use of tracking devices. Utilizing a drone, however, does not come with the same levels of upfront and recurring seasonal costs. For instance, the UAV used for previous work was the DJI Mavic Enterprise 2 Dual, further described in **Section 4.4.1**. It is no longer in production, but its successor, the DJI Mavic Enterprise 3, is available at approximately 44,000 Norwegian kroner [32]. Even though the practical use cases of the UAV and radio bells are not entirely the same, comparing these numbers indicates the possible savings and value associated with using UAVs for sheep monitoring and roundup.

3.2 Object Detection

Object detection is a computer vision technique for locating and classifying instances of objects in images [33]. An object detection algorithm uses an image as input and outputs bounding boxes predicting the object's location and a classification prediction with a confidence score, stating the algorithm's confidence in its prediction. An example of input and output is shown in **Figure 3.1**. The algorithms used for object detection use machine learning or deep learning to detect and classify objects correctly. Object detection has many use cases, from facial recognition or autonomous driving cars to surveillance, security, or healthcare [34]. Object detection is also a basis for other computer vision techniques, like instance segmentation and object tracking.

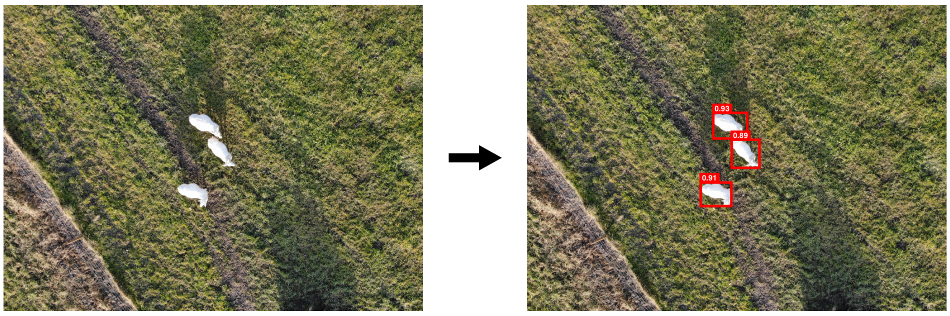


Figure 3.1: Example of object detection input and output

3.2.1 Evaluation Metrics for Object Detection

This section provides an overview of evaluation metrics commonly used for object detection, such as precision, recall, F1-score, and mean Average Precision (mAP). The intricacies of these metrics, their calculation, and their interpretation in the context of object detection will be described. These metrics are standard for evaluating object detection algorithms' performance.

Intersection Over Union

Intersection over union (IoU) is a metric that evaluates the overlap between two bounding boxes, for example, how well a prediction is relative to the ground truth [35]. IoU for two bounding boxes are measured by the intersection area divided by the union area, as shown in **Figure 3.2**.

By calculating IoU, a prediction is classified as a True positive (TP) or False positive (FP). The prediction is classified as correct and a true positive if the IoU exceeds a predefined threshold, typically set to 0.5. A prediction falling below the IoU threshold is classified as incorrect and a false positive. A False negative (FN) means that the object detector didn't make a prediction and missed a ground truth. A True negative (TN) means no prediction or ground truth exists, which

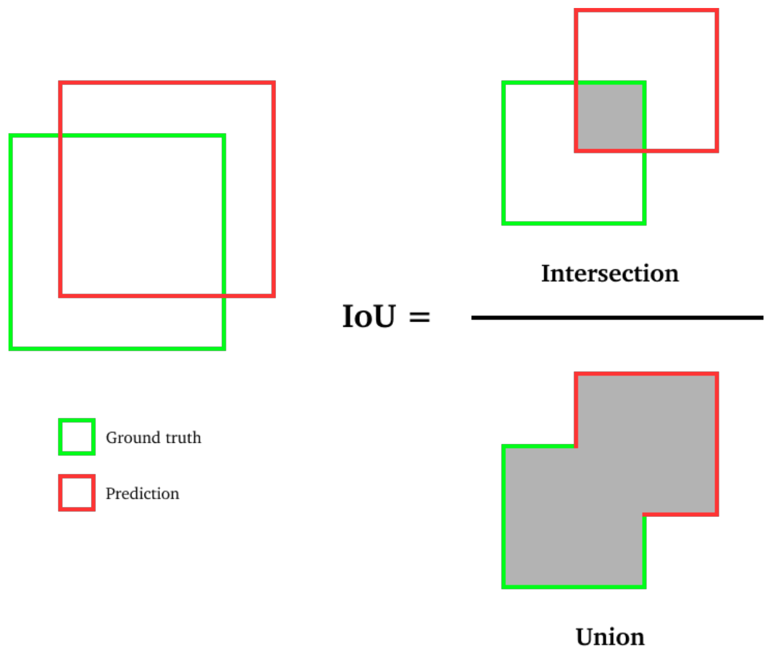


Figure 3.2: Intersection over union (IoU)

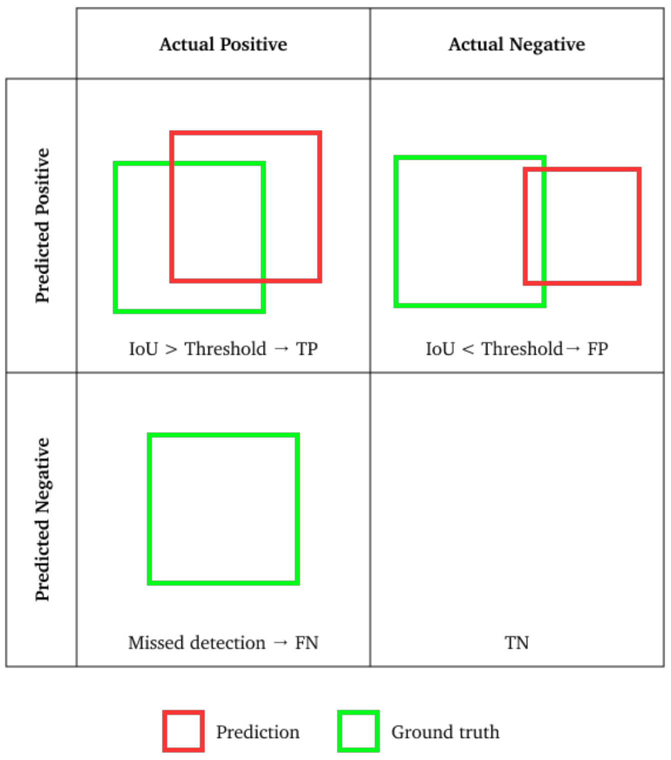


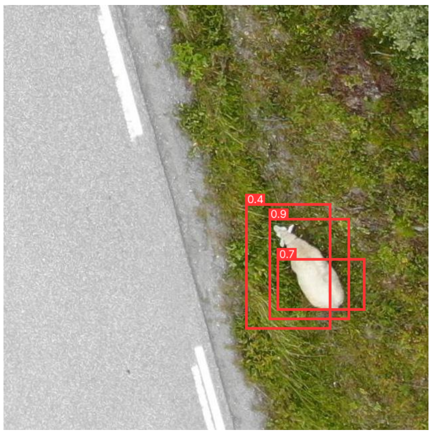
Figure 3.3: Confusion matrix

typically are not recorded when reporting predictions performance. The *confusion matrix*, as illustrated in **Figure 3.3**, provides an overview of these classifications, summarizing the performance of the detection algorithm.

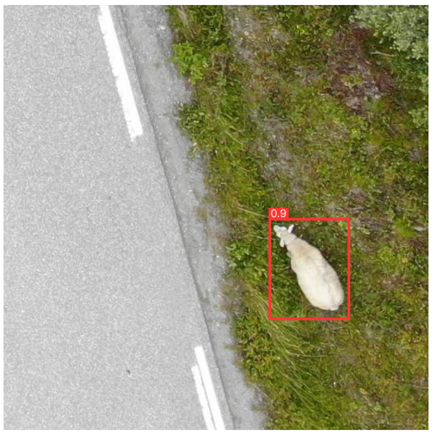
The selected IoU threshold will decide how accurate the bounding box is required to be relative to the ground truth. A low IoU threshold will result in more inaccurate predictions being considered true. Consequently, a higher IoU threshold will result in only precise bounding boxes being considered true.

Non Maximum Suppresion

Non maximum suppresion (NMS) is a post-processing algorithm widely used in computer vision. When an object detection model predicts where an object is, it usually makes several predictions that overlap for the same object. Each prediction has a confidence score, and only the prediction with the highest confidence is useful for this object. To find it, all the predictions are sorted in descending order in a list of potential detections. Then the prediction with the highest confidence is added to the final list of detections. Then it removes any other predictions in the potential detections list with an IoU over a pre-defined threshold. This continues until the list of potential detections is empty [36]. Figure 3.4 shows an image before and after applying NMS.



(a) Before NMS



(b) After NMS

Figure 3.4: Bounding boxes before and after non-maximum suppression

Precision and Recall

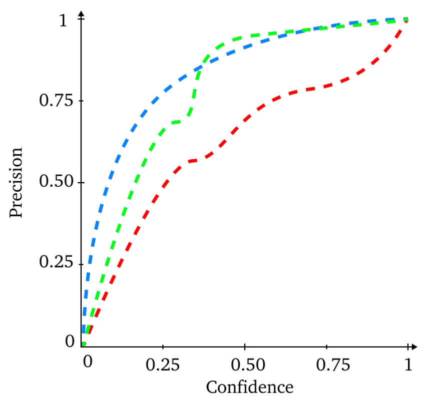
Precision and recall are metrics that describe the performance of the predictions of an object detection model. Precision is expressed by **Equation (3.1)** as the fraction of correct prediction out of all predictions [35]. In this case, the fraction of predicted sheep that actually are sheep. Precision will be a number between 0 and 1. If precision is closer to 0, it would mean that most of the model predictions are FPs, meaning the predictions are wrong. Similarly, a precision score close to 1 would mean a high degree of TPs and most predictions being correct. **Figure 3.5a** exhibits possible relationships between precision and the confidence threshold. A low confidence threshold would result in many predictions where most would be wrong, resulting in a low precision score. As the confidence threshold rises, fewer, more confident predictions are made, resulting in a high precision score.

$$precision = \frac{TP}{TP + FP} = \frac{\textit{All correct predictions}}{\textit{All predictions}} \tag{3.1}$$

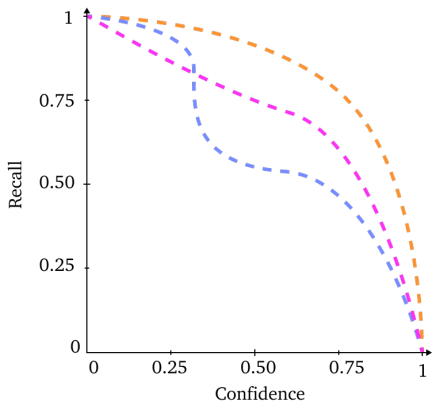
Recall is expressed by **Equation (3.2)** and is defined as the fraction of correct prediction out of all correct instances [35]. In this case, the fraction of sheep correctly predicted out of all instances of sheep. Like precision, recall is a number between 0 and 1. A recall score close to 0 would mean many FNs, meaning it does not find the relevant instances that it should. A recall score close to 1 would mean a high degree of TPs and low FNs, meaning it would find most sheep. **Figure 3.5b** exhibits possible relationships between recall and the confidence threshold. A low confidence threshold would result in many predictions, making it likely that most ground truths are found, resulting in a high recall score. As the confidence threshold rises, fewer predictions will be considered true, resulting in fewer ground truths being found and a lower recall score.

$$recall = \frac{TP}{TP + FN} = \frac{\textit{All predictions}}{\textit{All ground truths}} \tag{3.2}$$

Neither precision nor recall is suited to be a single metric of accuracy, and they need to be reported together to describe the models' accuracy adequately. The relationship between precision and recall is exemplified in **Figure 3.6**. Precision does not consider how many total ground truths it missed. That means the precision of a model can be 100% if all the predictions it made were correct, even if it missed 80% of the total ground truths. Meanwhile, recall does not consider how many predictions were made, only if all ground truths were detected. That means the recall can be 100% if the model predicts everything as a ground truth, even if that would be detrimental to the precision. They are both influenced by the confidence threshold but have the opposite relationship. A low threshold gives low precision and high recall, while a high threshold gives high precision and low recall [37].

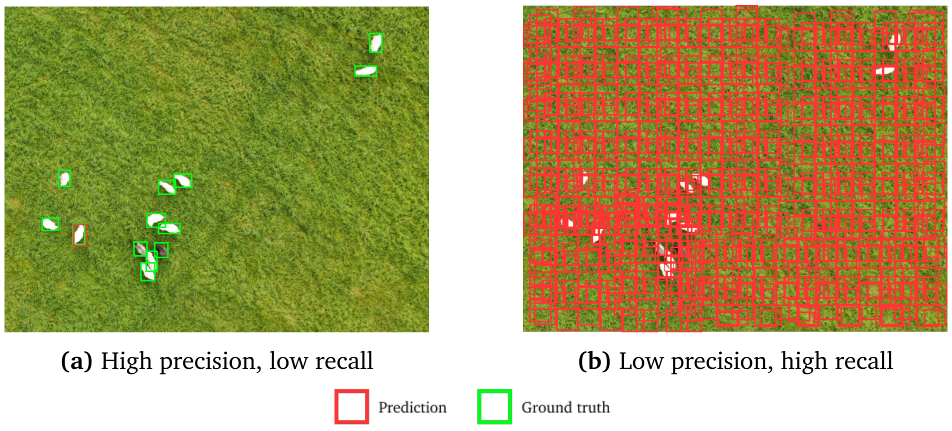


(a) Precision vs confidence



(b) Recall vs confidence

Figure 3.5: Possible relationships between confidence threshold and precision or confidence



(a) High precision, low recall

(b) Low precision, high recall

Figure 3.6: The relationship between precision and recall and how they affect each other

F1-score

F1-score is the harmonic mean of precision and recall and is expressed by **Equation (3.3)**. It is a weighted single metric score combining precision and recall [38]. The F1-score is between 0 and 1, where the precision and recall are weighted equally. Plotting the F1-scores of multiple confidence thresholds creates an F1-curve, where the curve's apex signifies the optimal confidence threshold for both precision and recall [39].

$$F1\text{-}score = 2 \times \frac{presicion \times recall}{precision + recall} \tag{3.3}$$

Precision-Recall Curve

Plotting the precision and recall across all confidence thresholds results in a precision-recall curve [35]. It describes the relationship between precision and recall and better represents accuracy than precision or recall alone. At a given recall value, it shows the corresponding precision, showing the tradeoff between the two metrics. Different potential precision-recall curves are shown in **Figure 3.7**. When recall is low, precision tends to be high, as few ground truths have been found, but the model is confident they are correct. As the recall rises, there need to be more predictions made to find more ground truths, which typically results in the models making more wrong predictions, decreasing the precision. A good-performing model would keep precision high overall recall levels.

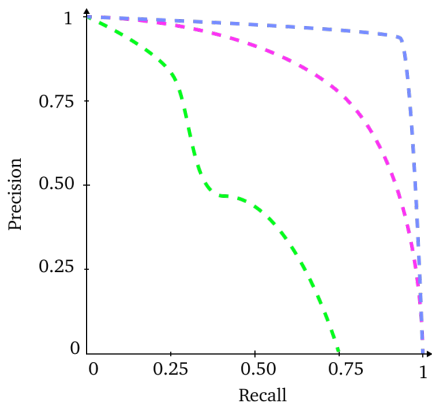


Figure 3.7: Possible precision-recall curve

Average Precision

Average precision (AP) is used to compound the precision-recall curve into one metric. The AP value is the area under the precision-recall curve *p(r)*, and can be found for each class detected using **Equation (3.4)** [35].

$$AP = \int_0^1 p(r)dr \tag{3.4}$$

Mean Average Precision

The mean of all average precision values is called Mean average precision (mAP) and is the more wieldy used evaluation metric since most object detection models have multiple classes. The mAP score of a single class model is the same as the AP score. **Equation (3.5)** shows the formula for mAP for every n number of classes.

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{3.5}$$

Two commonly used mAP versions are mAP@0.5 and mAP@0.5:0.95, where the number indicates the IoU threshold used for calculations. mAP@0.5 describes the mean average precision at 0.5 IoU threshold [35]. Similarly, mAP can be calculated at other thresholds, like mAP@0.75. mAP@0.5:0.95 describes the mean average precision over multiple IoU thresholds from 0.50 to 0.95 with a 0.05 increase each step [35]. By doing this, the inaccurate bounding boxes will be regarded as false as the IoU threshold rises, indicating how tight the model's bounding box predictions are. mAP@0.5 and mAP@0.5:0.95 are usually reported together to evaluate a model's performance.

3.3 Artificial Neural Networks

Artificial neural network (ANN) is a collection of nodes and edges organized to simulate the structures and connections of neurons in the brain [40]. It is structured with multiple layers, the input layers, a number of hidden layers, and the output layers. **Figure 3.8** shows an example of a simple fully connected artificial neural network [41].

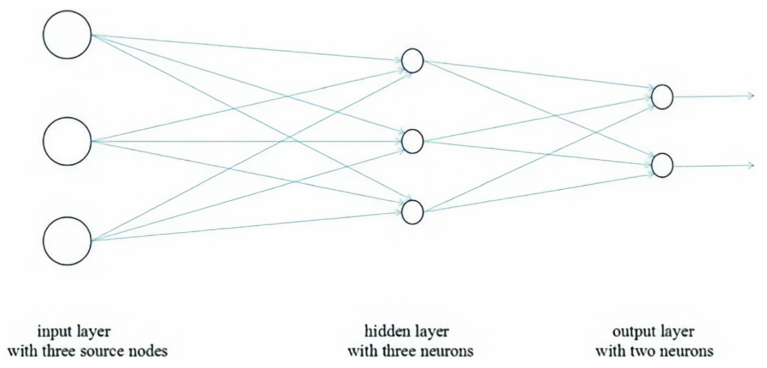


Figure 3.8: Example of a simple fully connected artificial neural network with input, hidden, and output layers. Figure from Zhang [40]

The input is initially processed in the network's first layer. It performs some computations before forwarding the result to the next layer. Similar calculations take place at each subsequent hidden layer in the network. Eventually, the output layer is reached, with the final results from passing the input through the network. The output is then compared with the desired or target output. The discrepancy between the actual and target output is called *loss* and is used to adjust the network parameters for more accurate results. The process of updating parameters is called *backpropagation*. The iterative process of input processing, comparison with the desired output, and parameter adjustment is continued with new inputs. This process is referred to as training the network [42].

Every neuron in a fully connected network has a number of input edges corresponding to the number of neurons in the previous layer. Each input edge has a corresponding *weight*. The neuron summarises all the weighted input from the previous layer and adds its own *bias* [40]. Before the calculation is passed to the next layer, a *activation function* is applied to prevent linearity [43]. A common activation function is the *Sigmoid* function which outputs either 0 or 1 if the value is above or below a certain threshold. The weights and biases in the network are the parameters which are adjusted during training. **Figure 3.9** shows an example of the input and output of a neuron.

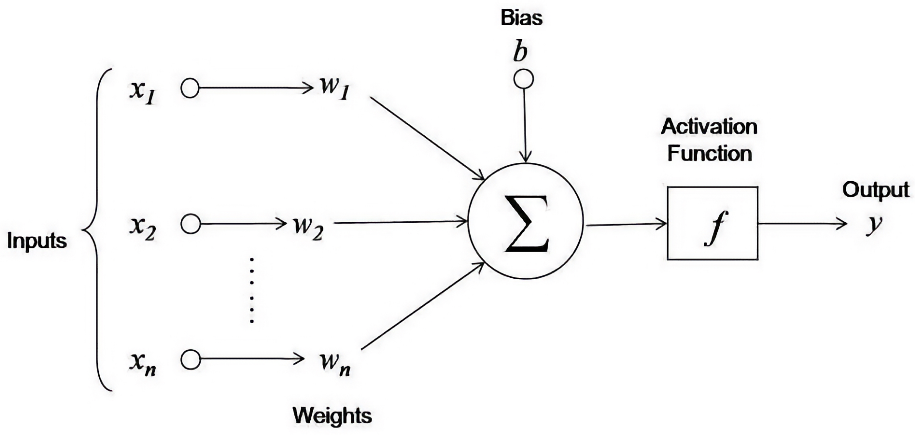


Figure 3.9: Example of a neuron from an artificial neural network. Figure from *Neural Networks and Machine Learning* [44]

3.3.1 Convolutional Neural Network

A Convolutional neural network (CNN) is a type of ANN which is often used for computer vision tasks [45]. A CNN consists of three building blocks, the backbone, neck and head. The different parts are made up of multiple layers. The backbone is responsible for creating a feature map from the input and applying different functions to the input. The neck is responsible for preparing the features extracted from the backbone to the head. The head is responsible for the classification and predictions and is usually a fully connected layer outputting a probability for the input to belong to a certain class. Through these three parts, CNN can accurately classify objects in images [46]. Figure 3.10 is an example of a CNN architecture.

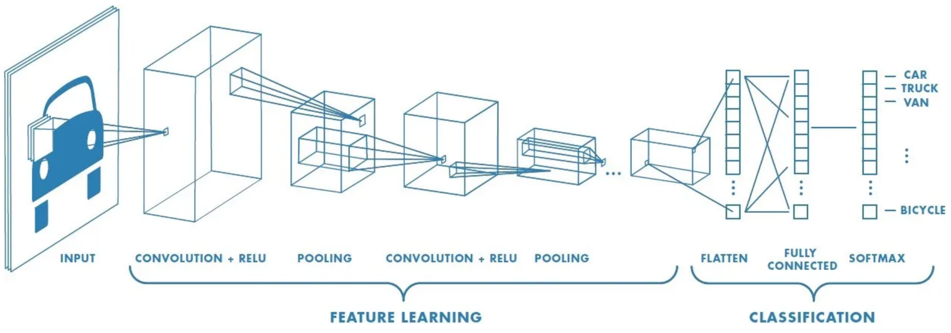


Figure 3.10: Example of a CNN architecture. Figure from Saha [46]

Convolution Filter

The convolution filter is used to extract desired features from the input. A filter is applied to subregions of the pixel grid representing the input image. The filter strides across the pixel grid and outputs a feature map highlighting the desired features of the input image. These features can, for example, be horizontal or vertical edges [46]. **Figure 3.11** shows a convolution filter being applied to an image.

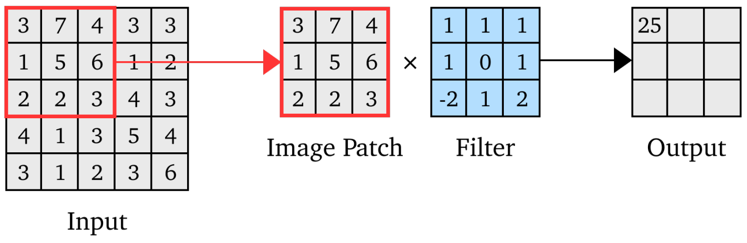


Figure 3.11: Convolution filter with 3x3 image patch and filter

Pooling Operations

Pooling is applying a pooling layer to extract the desired features from the input. Max-pooling, shown in **Figure 3.12** is one such pooling operation that reduces a region of the feature map to the highest value from the region which is beneficial for computing costs [47]. Other pooling operations are average pooling, which averages out the values of a subregion to reduce noise, or min-pooling shown in **Figure 3.13**, which extracts the lowest values [48].

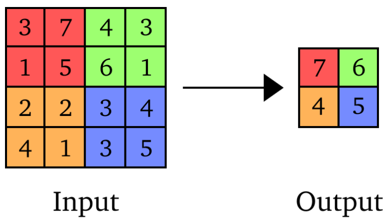


Figure 3.12: Example of max-pooling operation on a 2x2 image patch and stride of 2

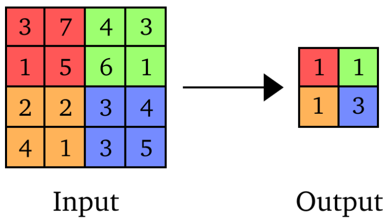


Figure 3.13: Example of min-pooling operations on a 2x2 image patch and stride of 2

3.3.2 YOLOv7

You only look once (YOLO) is a family of real-time single-stage object detection algorithms. The algorithm inputs an image and outputs the class's predicted bounding boxes. The first version, YOLOv1, was released in 2016, and since then, multiple new versions have been proposed improving upon speed and accuracy. The latest official version is YOLOv7, which was released in 2022 and is a state-of-the-art object detector [49].

The YOLO algorithms are built to be a trade-off between speed and accuracy. They are single-stage, meaning they localize objects in images in one step, compared to multi-stage detectors, which propose a general area before locating the actual object. [50]. Single-stage detectors have better inference speed than multi-stage but at the cost of accuracy [51].

YOLOv7 is a CNN consisting of a backbone, neck and head. It uses the RepConvN backbone and YOLO head [52]. The Extended efficient layer aggregation network (E-ELAN) computational block is introduced in YOLOv7, which uses group convolution to enhance the features learned by different feature maps and improve the use of parameters and calculations [49].

When evaluating the loss of the model, YOLOv7 used three different loss functions, objectness loss, box loss and classification loss [53]. Objectness is the probability that an object exists in a defined area. Objectness loss is the error in the confidence of the presence of an object. Box loss is the error between the ground truth and the predicted bounding box. Classification loss measures the error in class prediction.

YOLOv7 Bounding Box Annotations Style

Each annotated image is accompanied by a corresponding *.txt* file that contains the labels. The label file provides information about the bounding box coordinates and class labels for each object within the image. Each line in the label file represents one object in the image. The YOLO label format is shown in **Equation (3.6)**.

$$\texttt{center\_id} \quad \texttt{center\_x} \quad \texttt{center\_y} \quad \texttt{width} \quad \texttt{height} \tag{3.6}$$

The `center_id` indicates the class of the object. The `center_x` and `center_y` represent the coordinates of the bounding box's centre relative to the image's width and height. These centre coordinates are normalized, ranging from 0 to 1. In this format, (0,0) represents the image's top-left corner, and (1,1) represents the bottom-right corner. The `width` and `height` represent the width and height of the bounding box relative to the image's width and height. These values are also normalized, ranging from 0 to 1.

3.4 Training a Neural Network

3.4.1 Gradient Descent

When training a neural network, the goal is to adjust the weights and biases to minimise the loss. The technique for finding the optimal values is called gradient descent [54]. In broad terms, gradient descent tries to find the optimal values by calculating the gradient for the loss, which tells which direction the steepest increase is. Adjusting the parameters in steps in the opposite direction of the gradient, that is, towards the steepest decrease in loss, the minimum loss is reached incrementally.

The gradient of the loss with respect to the weights and biases is calculated by initializing random values for weights and biases. For each training example, the algorithm calculates the gradient of the loss and updates the weight and biases by subtracting a fraction of the gradient from the current values. This fraction is called the learning rate and determines the step size toward the steepest decrease. The process is repeated for multiple iterations, called epochs, until the algorithm converges to a minimum of the loss function. Figure 3.14 shows the process of finding the minimum loss with gradient descent.

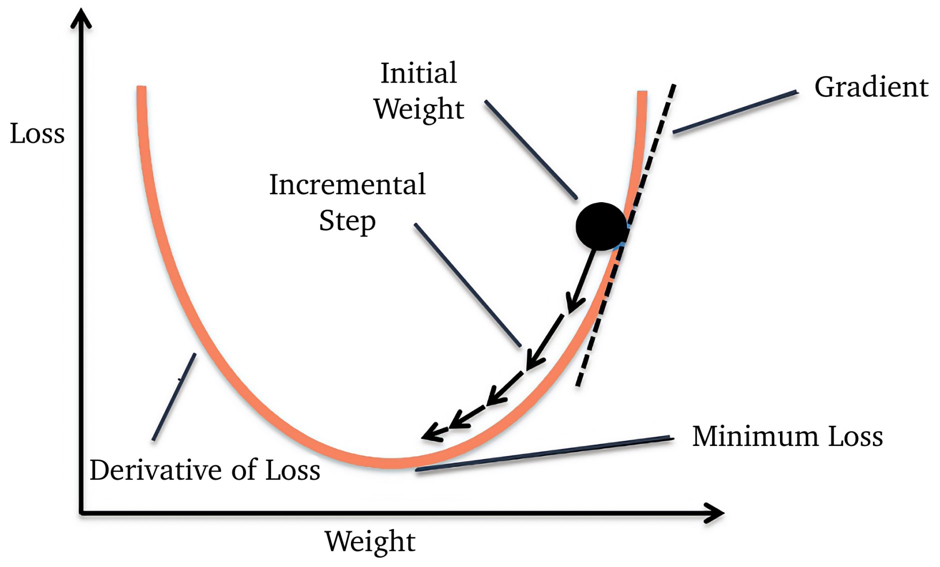


Figure 3.14: Gradient Descent. Figure adapted from Crypto1 [54]

3.4.2 Supervised vs Unsupervised Training

Supervised and unsupervised learning are two different approaches when training models in machine learning [55]. Both approaches have their unique benefits and drawbacks.

In supervised learning, a labelled dataset is used to train the model, meaning that both the input data, in this case, the images, and the desired output, the bounding boxes and annotations, are provided during training. CNNs use supervised learning technique and has proven to be highly effective and accurate for object detection, like with YOLOv7 [49]. However, they require a significant amount of labelled data, which can be time-consuming and costly.

In unsupervised learning, a labelled dataset is not required. It tries to identify patterns and features from the input data, like specific shapes, textures or colour combinations. Unsupervised methods work with unlabeled data and can be easier to deploy at scale. However, accuracy is often lower than with supervised techniques, especially in tasks requiring precise object localisation. It can be useful in exploratory data analysis or when labelled data is scarce.

3.4.3 Pre-training and Fine-tuning

Pre-training is a technique in machine learning about training a model without any specific task in mind. It is done on a large dataset like MS COCO or VOC. Pre-training aims to learn generic features, like shapes or textures, that can be used for other more specific tasks [55]. When a model is pre-trained on a dataset, it can be used for object detection, but to make it accurate on a specific task, fine-tuning can be used to train the model further. In fine-tuning, the model is trained on a smaller dataset with image labels for a specific purpose, such as recognizing sheep [56].

The model can perform better using pre-training and fine-tuning than training from scratch [55]. With a dataset limited by size and the number of instances, this technique is especially valuable, as the limited number of instances may limit the models' ability to learn general patterns. By learning general features on a large, diverse dataset, the model can perform better at new and unseen data and improve its convergence speed.

3.4.4 Over- and Underfitting

Over and underfitting in machine learning are problems that can affect the model's performance by either emphasising too much on small details or not learning general features enough [57]. Overfitting is when the model learns the specific details and specifics of the examples in the training data instead of the generic features. An overfit model makes it bad at recognizing new variations of a class. Underfitting happens when the dataset used for training is limited in size and variation so that it does not learn the features for recognizing the objects.

3.4.5 Dataset Splits

When training an object detection model, the dataset is split into three subsets, train, validation and test. The split is essential to ensure the model has sufficient data for training while still having new examples for validation and testing [58]. Typically, the training subset consists of 70% to 80% of the total dataset, while the validation and test subsets split the remaining images equally.

The training subset is the largest and should be varied and representative of the real-world objects it is being trained to recognize. This dataset is used for all epochs during training.

The second subset is the validation dataset. After each training epoch, the model is evaluated with the validation dataset to evaluate the model's performance. Comparing the models' performance on the validation dataset can help prevent overfitting while training. If the performance on the training dataset is good but bad on the validation set, it can be a symptom of overfitting, as it doesn't generalise well on new unseen data[58].

The third subset is the testing dataset used to evaluate and compare performance results. These images must be unique for this dataset to evaluate the performance of the trained models accurately [58].

3.4.6 Datasets for Object Detection

Different datasets have been created for training and evaluating object detection models. These datasets have different attributes, making them useful for computer vision tasks. An important aspect of the datasets is that they are benchmarks for new and improved object detection algorithms. As a new algorithm is released, it is tested on a recognized dataset like MS COCO, so there is a common dataset to test and compare performance. Algorithms like YOLOv7 are tested and compared to other object detectors on the MS COCO dataset [49].

MS COCO is created by Microsoft [59] and consists of 33,000 images of objects from 80 different classes, ranging from cars, bicycles and animals. Each image in the dataset is labelled with annotations for the object classes. The dataset is used in computer vision tasks like object detection and segmentation. It is used for training and evaluating object detection models and has become a benchmark in the industry.

The Visual Object Classes dataset is another dataset used for computer vision tasks [60]. It was first introduced in 2005 for the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) Visual Object Classes Challenge, a computer vision competition. In later years, the dataset was updated for new challenges like the VOC2012. The dataset consists of 11,500 images and 20 classes, making it inferior in size to MS COCO, but the dataset still has some value as a benchmark for new computer vision algorithms.

Chapter 4

Method

The following chapter will describe the method used for answering the research questions. It will give an overview of the requirements for the synthetic images and object detection models' performance, describe the method for generating synthetic images, image preparation of real image data, and explore alternative solutions.

4.1 Requirements

Certain requirements must be in place to ensure the usefulness of the object detection model and the synthetic images. This section will cover requirements for synthetic image generation and requirements for object detection models' performance.

4.1.1 Requirements for Generated Images

Generating synthetic images for machine learning applications has proved to be an effective way of supplying a dataset [15][13][23][24]. However, these methods are not perfect, so factors like the domain gap [21] must be considered. This section will present functional and non-functional requirements for synthetic image generation.

Functional Requirements

When generating synthetic data, key properties must be fulfilled to make the images functional and practical for sheep detection. Firstly, the synthetic images must be photo realistic to a certain extent. This might seem fairly oblivious, but to decrease the domain gap, they must be realistic looking enough to be useful when applying the learned features to real images. Secondly, it must support the creation of random objects, scenes and layouts. The scenes useed for training must be varied to a certain extent and randomly created within certain pre-defined parameters. This is important to ensure variation in the training images to prevent overfitting. As well as supporting random layouts, randomised low-level features are important for sheep objects and the landscape. The sheep objects should have random sizes and colours within a predefined range, and the landscape must contain vegetation of different sizes, colours and appearances. As the primary focus is occluded sheep, placing objects to create the specific occlusion conditions must be controllable and not completely arbitrary. The sheep should be placed close to the border of other objects, like trees and bushes, which also should be randomly positioned. Weeding out images without occluded sheep in post-processing is not viable as it would require too much computing power and manual work, meaning all images generated must contain an occluded sheep. All images must also have precise labels, as labelling images is tedious and requires much time and would render the generation method of low value.

Non-functional Requirements

For an object detection model to be sufficiently trained, having loads of data to train on is crucial. This requires the method to support the generation of large batches of images. It must be able to do so without supervision, and the images must have the required properties regarding randomization and variation. The software for generating images must be able to run on a 16-inch MacBook Pro with the Apple M1 chip. It cannot be too graphics-heavy, as access to graphical computing power for image generation is limited. The size of the images should also be minimal, as the available storage capacity is limited. Within these limits, generating the images within a reasonable timetable should also be possible.

4.1.2 Requirements for Object Detection Model

Certain requirements exist for the model's performance to be a viable option in the context of sheep roundup. However, as the thesis doesn't focus on running the models on the UAV, the performance requirements are somewhat different. The focus will be on accuracy performance and what is required for the model to be useful for the farmer in the field. Inference speed and computational power are out of the scope of the thesis and not considered for the requirements.

The drone will be used in the final phase of the sheep roundup. It will not be used to scan through the entire pasture but rather as a tool to locate the final sheep. As the farmer is searching, he might have an idea or know roughly where to look. He can then use the drone to check out the potential location of the sheep. While flying in a pre-defined pattern over the search area, the UAV must indicate if it finds something so the farmer can check it out more closely. Because an indication for further search is the UAV's main purpose, recall is valued over precision for the specific use case of the object detection model. Simultaneously, there must be an adequate level of precision as well, as it would be a waste of time for the farmer to search through flagged areas to find the UAV detected a rock. Therefore, the trained models for the thesis require high levels of precision and recall, but recall is prioritised over precision if needed.

4.2 Unity

Based on the requirements for the generated images in mind, the game engine Unity was chosen to generate the synthetic images. Unity is a game engine developed by Unity Technologies written in C++ [61]. Unity makes it possible to create photorealistic scenes and controllable environments using C# without too heavy hardware requirements. Unity is widely popular for game development, and there is a large community of developers and support online. Assets are an important part of the engine and the objects used to create a scene. There are a lot of open-source and paid resources available online. The assets are of varying quality but can come close to photo realism. The environments are fully controllable, and the object's behaviour is highly predictable, making it suitable for creating scenes and objects for synthetic image generation.

4.2.1 Unity Perception

With Unity, the package Unity Perception was used. Unity Perception is a tool created for large-scale creation and generation of synthetic images for object detection tasks. It has tools for randomizing, generating and annotating images automatically. These tools can generate images with randomised properties, both built-in and customised, including camera, lighting, scenes and landscape. It has proven to be an effective tool for synthetic image generation [24][13].

Limitations in Unity Perception

A limitation when using Unity Perception is how it handles the generation of bounding boxes for objects occluded by transparent and opaque objects. For sheep under opaque objects, the bounding boxes will only be drawn for the visible parts of the sheep, despite the fact that the bounding box should also be drawn for the occluded parts of the sheep. With transparent objects, however, it does not account for what is possible to see through and not. For instance, with a simple mesh covering the sheep, like leaves on a branch, it can be possible to see the sheep through the mesh. Still, Unity Perception cannot properly label the visible sheep behind the transparent mesh while at the same time excluding those that are completely covered and not visible. The Unity Perception team plans a fix for these issues, but not available at the time of writing this thesis. As occluded sheep is an integral part of this thesis, a workaround must be implemented. In earlier work, the images of sheep not visible underneath the tree were removed manually from the dataset [13]. As the generation of images must be scalable, this is not a viable solution for larger batches of generated data.

Another limitation regarding Unity Perception is its compatibility with MapMagic 2, used for generating the landscape in synthetic images. When generating images with Unity Perception and MapMagic 2, the generation of the scene and capturing of the image must be synchronised. MapMagic generates terrain asynchronously from Unity Perception, resulting in the terrain not being ready as the image is captured, resulting in an empty image. To account for this, each iteration of image generation has followed a process frame-by-frame, described in **Section 4.3**.

4.3 Data Generation in Unity Perception

For the requirements described in **Section 4.1**, Unity Perception is a suitable tool for generating synthetic data. However, the existing limitations need a workaround. The following section describes the adjusted step-by-step process for generating occluded sheep data. The steps can be found in **Table 4.1**.

Frame	*Action*
0	Load terrain Load and position tree
1	Buffer
2	Load sheep Load sheep animation
3	Place sheep Get sheep mesh Change color Get bounding box Generate bounding-box file Capture image
4	Remove sheep Remove tree
5	Buffer

Table 4.1: Frame-by-frame process for generating images with Unity Perception

Initialization

In the initial stages of image generation, the steps involve loading assets and preparing the scene. The terrain and tree are created and loaded onto the scene in **frame 0**. To use the tree's position to occlude a sheep, it must have a *collider*. A collider defines a game object's shape and size for physical collision. Different types of colliders have varying degrees of resemblance to the object's shape. The degree of resemblance can be adjusted according to the use case. If the interaction between objects doesn't require precision, a geometric shape like a box or a cylinder could be used as a collider. Likewise, an accurate collider makes it possible to have precise interactions between objects. This comes at the cost of computing power, as it is much more efficient to calculate the collision between two moving boxes than two moving complex shapes with lots of vertices. Using a *mesh collider*, the exact shape of the tree is defined and made possible for collisions. The target is to have a precise collider so that the exact shape of the tree branches and leaves can be used. After one buffer frame, **frame 1**, the tree and collider are fully loaded.

Sheep Positioning and Image Capture

In **frame 2**, the process of sheep positioning starts. The sheep and its animation are loaded onto the scene. As the asset is loaded, **frame 3** starts. A mesh is generated with the current frame of the animation, as described further in **Section 4.3.2**. For a sheep's position to be correctly considered occluded, the sheep must be between 5 % and 95 % covered by leaves, branches, or other parts of the tree. To calculate the percentage that the sheep is covered, a *linecast* is used. A linecast is a Unity function that creates a line from a point of origin to a destination point. The linecast returns whether or not the line reaches its target destination. Using the linecast function, it is possible to determine if the positioning of the sheep is under the tree and to what degree it is covered.

A number of lines are set from the camera to the sheep's position. The lines are set to cover each of the sheep's mesh vertices. Since the only collider in the scene is the mesh collider for the tree, it will act as a layer mask, covering the sections of the sheep covered by the tree. By calculating the number of lines that successfully reach the sheep, it can be determined whether the tree occludes the sheep to the required degree. The process is reiterated if the sheep are not covered extensively. That could happen if the sheep is positioned far from the tree, is entirely visible, is completely covered, or is covered but not to the required degree. The positioning of the tree and sheep is random. Because of this, the tree can be positioned to cover the entire field of view for the camera, meaning there is no way for the linecast to hit the sheep. After 100 tries, the algorithm breaks to avoid a continuous loop, then repositions the tree before again trying to position the sheep. An example of the sheep positioning process is shown in **Figure 4.1**

As the sheep positioning is complete, the bounding boxes are generated using the same sheep mesh. They are generated by finding the outer vertices of the sheep. The maximum and minimum values in the x direction and the maximum and minimum values in the y direction are found. Using Unity's camera function *WorldToScreenPoint*, the x and y values can be transformed into points in the image. These values are used in the **Equation (4.1)** to calculate the values needed for the annotation style shown in **Section 3.3.2**. Then the bounding box label file is generated and filled with the bounding box values. At the end of this frame, the image is captured of the scene. The results are an image of a partially occluded sheep with the corresponding label for that sheep.

$$\begin{aligned}
\textit{Object width} &= \frac{\textit{x maximum} - \textit{x minimum}}{\textit{image width}} \\
\textit{Object height} &= \frac{\textit{x maximum} - \textit{x minimum}}{\textit{image height}} \\
\textit{Object center x} &= \frac{\frac{\textit{x maximum}+\textit{x minimum}}{2}}{\textit{image width}} \\
\textit{Object center y} &= \frac{1 - \frac{\textit{x maximum}+\textit{x minimum}}{2}}{\textit{image hight}}
\end{aligned} \tag{4.1}$$

Cleanup and Repeat

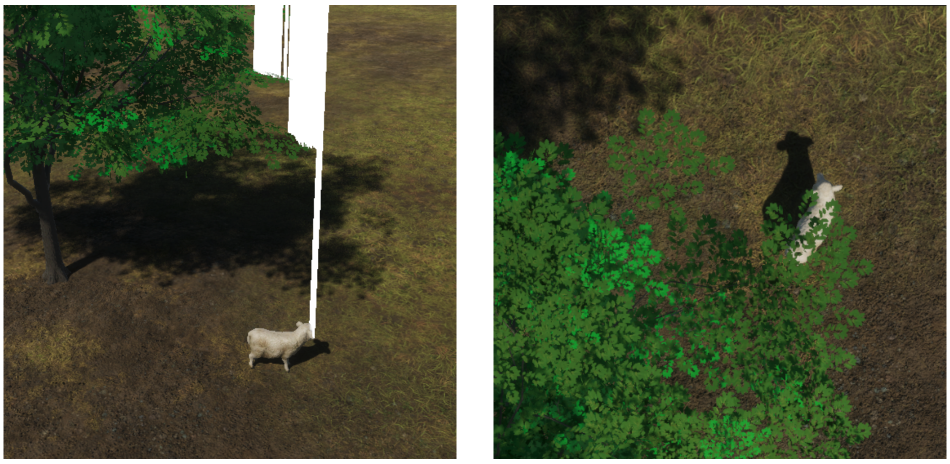
In **frame 4**, the scene is prepared for a new iteration as the image has been captured. The tree and sheep are removed before a buffer frame, **frame 5**. The buffer frame is in place so that the tree mesh collider is removed properly, as Unity colliders are on a 1-frame delay. After the last frame, the iteration has ended. Then the process is repeated for the desired number of times. The result is a dataset of the desired size with occluded sheep ready for machine-learning tasks. **Figure 4.2** are examples of the images generated.

4.3.1 Parameters and Considerations for Synthetic Image Generation

There are many considerations when generating synthetic images to maximise value for sheep detection. Unity Perception makes generating batches of randomised images possible, so the scope and confines of what to generate must be considered to maximise usefulness within the requirements.

As the focus of generating synthetic images is to supply the dataset with more instances of the occluded classes, it is crucial to define what occluded signifies. With an explicit definition, the algorithm for generating data can determine which images are accepted and which are declined. If the threshold for accepting the sheep in an image as occluded is too low, it will result in images with sheep that are not covered to the required extent, meaning the object detection model will not learn how the features of the sheep will look if they are covered. A too-high threshold will result in generated images where the sheep is close to fully occluded by leaves or branches, meaning the model has a very limited amount of meaningful features to learn, if any. Calculating the degree of occlusion is described in **Section 4.3**. After testing with multiple thresholds, a coverage lower than 5% was considered the minimum, and a higher coverage than 95% was considered the maximum.

The resolution of the images was selected to be 640px by 640px. The YOLOv7 architecture has a base native resolution of 640x640 which fits nicely with the image generation output. With a larger image size, it would be necessary to tile or



(a) Linecasts drawn from the camera position to the vertices of the sheep

(b) Occluded sheep from above, as seen in final image

Figure 4.1: Determining if a sheep is positioned under a tree

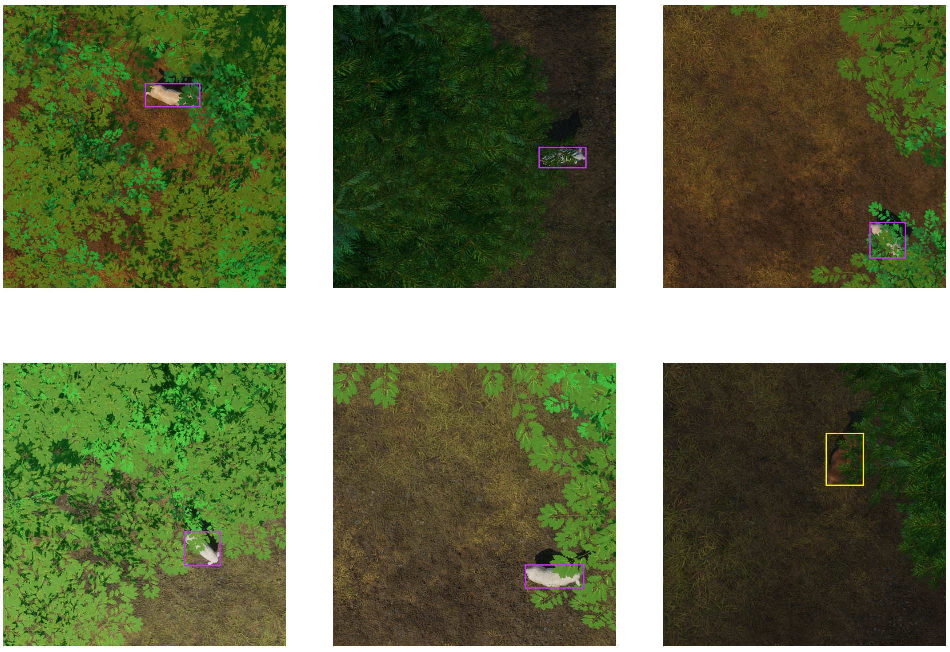


Figure 4.2: Resulting synthetic images of occluded sheep generated with Unity Perception

downscale the generated images as is done with the real images. Downscaling has proven less effective than tiling in performance [17]. Even though the resolution is low, it doesn't affect the clarity of the images, as the landscape and objects in the image are big enough to be sharp.

There are many different options for how to compose a scene. One option is creating large landscapes with lots of assets populating them. These scenes look realistic and true to nature. A camera setup could cover most of the scene, quickly capturing multiple images in succession. Another option is to have smaller localised scenes and images. There is less space available in the scenes for many assets, but this format is useful for focusing on specific details, like occluded objects.

4.3.2 Randomisers

Different randomisers have been used to create variations within the generated scenes. This section will briefly overview the randomisers and how they are configured.

Terrain

MapMagic 2 was used to generate random landscapes with desired features. MapMagic 2 is a Unity package that generates infinite landscapes from predefined settings. MapMagic uses a node-based logic system to generate the world's terrain, where each node represents a separate algorithm called a generator [62]. A generator can, for example, be noise, erosion, curve or blend. A landscape with the selected nodes is created by chaining different generators together. By defining parameters for pseudo-randomization and a random seed, MapMagic 2 outputs a randomised landscape within the predefined limits. The landscape for the synthetic images is recreated from the real datasets explored in **Section 4.4.2** as best as possible. MagMagic nodes are used to combine soil and grass texture, and combined using the randomiser.

Light

The scene has a single light source, acting as the sun, with the angle randomised using the inbuild *sun-angle randomsizer* and a light randomiser using the inbuild *light randomsizer*. It inputs the latitude position, the time of data and the day of the year. Using these inputs, it can generate correct lighting for a real-world position. As described in Section 4.4.2, most images were captured in the afternoon, and many were from the autumn. That means the sun will be lower in the sky. As *Vanherle et al.* [15] found, high-level features, like lighting in synthetic images, are essential to make them effective. The images were generated with latitudes between 50 and 70, with different hue levels, intensity and temperatures in the light. The day of the photo was set to be between September and November, and the time of day from 12 to 20.

Camera

The images are generated with a camera view looking straight down at the scene. Other options are possible, as the camera on the UAV has a field of view of approximately 85°, which can make some of the sheep in the images appear from the side. However, it was not prioritised as occlusion mainly occurs when the images are captured from straight above the sheep. The images are taken at five heights, 30, 40, 50, 60 and 70 meters. The height will influence the perceived sizes of the sheep in the images, accounting for different sizes of sheep in the images.

Trees

The trees generated in the landscape are from two premade assets from Unity's assets store [63][64]. There are 5 different oak and 16 different pine trees. Per the image, one tree is generated and placed in a random location in the landscape with an equal chance of being an oak or pine tree.

Sheep

The real images show that the sheep have different colours and sizes and stand in different poses. To mimic these features in the synthetic data, the sheep can have a range of colours. They are white, grey, brown and black, with different variations. The colours are created using a fur texture, which gets randomised colours from a noise generator. The noise creates variation in the fur, so the sheep get some random features and variation. Creating realistic-looking fur and texture for the sheep can be elaborated upon further. However, as low levels don't increase performance to a great extent [15], it was deemed adequate. When spawned in the landscape, the sheep gets a random colour from the set of colours available.

The sheep from the random images appear in a range of poses. The sheep asset used [65] supports a range of poses out of the box, from standing still, grazing, running, and laying down. As the sheep spawn, one of these poses is randomly selected for the sheep. As the image is captured, the sheep are set in a random frame from the animation, creating variations in the shapes of the sheep, which have shown to be important for performance [15].

4.4 Sheep Dataset

The dataset used for object detection in this thesis consists of real and synthetic images. These are captured and created for previous master theses by students at NTNU. This section will overview the different datasets, their classes and class distribution.

4.4.1 DJI Mavic 2 Enterprise Dual

The DJI Mavic 2 Enterprise Dual was used to capture the real images, shown in **Figure 4.3**. It is a remote-controlled UAV capable of being used for professional-grade applications like search and rescue, surveillance, agricultural inspections and other industrial applications. An overview of the specifications is listed in **Table 4.2**. The most important feature for the application of sheep images is the 12-megapixel RGB camera, producing 4000px by 3000px sized images.

The UAV is equipped with an IR camera, able to detect infrared light. The usefulness of IR images has been studied with varying results. Johannessen [8] and Furseth and Granås [17] results indicated that IR images could have a positive effect. However, Stemshaug [18] found that combining RGB and IR images to detect sheep hurt the performance of the object detectors. The exact impact of IR images on the effectiveness of object detection remains unclear due to these conflicting findings. Therefore, the IR images have been discarded for this thesis.

4.4.2 Real Image Data

There are 2125 UAV images in the dataset used for this thesis. These images have been captured in different years, locations, and seasons. The real images are central when training the object detection model for mixed training and fine-tuning [15]. They are also central when generating synthetic data. By studying these images, features and characteristics from the environment can be extracted and used for synthetic image generation.

The first images were captured in Storlidalen in the autumn of 2019, then later in Klæbu and Orkanger in 2020, and Holtan in 2021. The small number of locations for image capture might be a limiting factor when training the object detector. Because of the limited location variance, the background and surrounding terrain might be simial for much of the dataset, resulting in the object detector not learning to generalize when facing other types of landscape.

Most images have been captured in the autumn, between August and October. A smaller number was captured in May and June, meaning most images are captured around the time sheep roundup is done. The time of day also influences the images. Most are caught in the afternoon when the sun is lower in the sky. The lighting for the synthetic images uses these parameters to generate light.

In an analysis done by Østtveit [13], he tried to find the elevation at which the images were captured. He used Global positioning system (GPS) elevation data to see how many meters above sea level the image was captured. He then cross-checked the GPS position with data from *Karvtverket* to find meters above sea level for that exact position. Using these data, he could plot the height of the UAV flying above the ground when it captured the image. According to these results, some of the data he used were unreliable, as some images were claimed to be captured 70 meters below ground. Removing the questionable data points, he found that most images were captured between 30 to 60 meters above the ground, influencing the synthetic images' camera height.

Figure 4.3: DJI Mavic 2 Enterprise Dual

DJI Mavic 2 Enterprise Dual Specifications	
Take-off weight:	899g (without accessories)
Dimensions:	Folded: 214mm X 91mm X 84mm. Unfolded: 322mm X 242mm X 84mm
Speed	72 kph
Range	8000 km
Flight Time	31 min
Internal Storage	Micro SD, 24GB Onboard
Battery	3850 mAh (heated)
RGB camera resolution	12 MP
RGB camera Field of view (FOV)	Approx. 85°
Sensor	Sony 1/2.3
Lens	f/2.8-f/3.8
Zoom	2x Digital 3x Electronic
Shutter Speed	Electronic Shutter 8-1/8000s
ISO Range	Video: 100-3200 Photo:100-3200 (manual)

Table 4.2: DJI Mavic 2 Enterprise Dual specifications

Table 4.3 shows the distribution of sheep classes in the real dataset. Each class represents a type of sheep, and the table shows the number of individual instances of that class and the number of images containing at least one instance of that class.

Class	*Instances*	*Number of Images*
White sheep	12,380	1,739
Grey sheep	3,515	1,150
Black sheep	2,362	1,017
Brown sheep	1,071	445
White occluded sheep	904	516
Grey occluded sheep	149	75
Black occluded sheep	40	37
Brown occluded sheep	10	7
Total	20,431	4,986

Table 4.3: Instances and number of images for each sheep class in the dataset of real images

Evidently, the dataset is quite imbalanced towards the white sheep class, which appears over 150% more than the class with the second most instances, grey sheep. These sheep classes are the most similar, and it can be challenging to draw the line between a white sheep and a light grey sheep during labelling, meaning the classes with light-coloured wool dominate the dataset. It is also quite clear that the occluded classes are underrepresented in the dataset, especially those with dark wool. There are only 1,103 occluded sheep compared to 19,328 non-occluded sheep. Only 5.40% of all instances are occluded. Because of the limited number of real images with occluded sheep, an object detection model trained on this dataset alone will most likely struggle with recognizing the occluded classes.

4.4.3 Synthethic Image Data

Table 4.4 shows the distribution of sheep classes in the synthetic dataset. As seen, the spread of the number of instances of the classes is more equal than in the real dataset. Of the non-occluded classes, the brown sheep is the most numerous. Of the occluded classes, the occluded white sheep has roughly twice as many instances as the rest. The white sheep randomiser described in **Section 4.3.2** has two sets of white wool which can be selected. Because of the flat chance that one wool is chosen over another, the class of white sheep is twice the size of the others.

The classes which are not occluded are data generated by *Østtveit* for his thesis in 2022 [13]. These images are rendered in Unity but without the focus on occluded sheep. However, they include instances of occluded sheep, but they are not labelled correctly. An occluded white sheep from that dataset is labelled as a white sheep. It will influence the results when testing for multiple classes but will

Class	*Instances*	*Number of Images*
White sheep	851	778
Grey sheep	1,644	1,373
Black sheep	2,401	1,860
Brown sheep	3,182	2,275
White occluded sheep	3,245	3,245
Grey occluded sheep	1,699	1,699
Black occluded sheep	1,672	1,672
Brown occluded sheep	1,588	1,588
Total	15,233	6,904

Table 4.4: Instances and number of images for each sheep class in the dataset of synthetic images

not make a difference for single-class testing. A single class will be used for the use case of the UAV. But for the sake of experimenting, multiple classes will be used so that it is possible to compare and see the results on performance.

4.5 Data Preparation

Tiling of Images

When training an object detector model on image data, resolution can pose a challenge. Even with supercomputers available, the time required and the cost-value of allocating the resources needed make it unviable. However, there are solutions to this problem. One option is the use of downscaling. With downscaling, the images are compressed into a smaller resolution, which is more manageable for the hardware. When the resolution of the image is close to the desired resolution, downscaling can be a viable option, without losing too much detail. However, in this case, the images from the UAV are 4000px by 3000px, which is far from optimal. Some of the sheep from the UAV images are very small, so downscaling these images would remove too much detail. Another option is tiling, where the original images are tiled into smaller images which can be processed during training. *Granås & Furseth* [17] found that tiling was effective for UAV images of sheep for accuracy and training time.

The image is divided into an 8x6 grid. Every 4,056px by 3,040px image is divided into 48 507px by 507px images. To ensure that no sheep are split between two images, each of these smaller images is padded with 64 pixels on every side. As a result, most of the padded images have dimensions 635px by 635px, except for the edge images, which may be slightly smaller since they do not receive padding on the outer edges. After the division and padding process, some of the smaller images do not contain any sheep. These images are discarded. After tiling and padding, the same sheep may appear on two smaller images. Since the dataset is divided into testing, validation and training before the tilling process, this doesn't interfere with the model's training process. **Figure 4.4** shows an example of the tiling prossess.

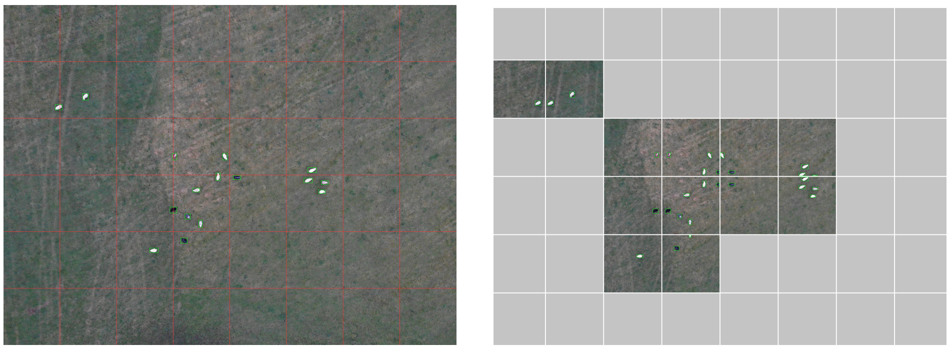

(a) Image divided into 8x6 grid **(b)** Images left after tilling and discarding

Figure 4.4: Example of image tilling

4.6 Alternative Approaches to Sheep Recognition in UAV Imagery

Before selecting the method of data generation and an object detection model, alternative approaches were considered. This section will give a brief overview of other viable options for exploring the goal of this thesis.

4.6.1 Exploring Methods for Synthetic Sheep Image Generation

To create synthetic images for sheep recognition in UAV imagery, Unity was chosen as the framework for data generation. However, alternative approaches exist that leverage different frameworks, varying from game engines to more specialized image generation techniques. By exploring these options and analyzing their strengths and weaknesses, the choice of approach can be better justified, and the overall methodology can be more thorough.

Exploring Unreal Engine as an Alternative

Another popular game engine which can be used for generating synthetic data is *Unreal Engine*, developed by Epic Games. Like Unity, it can render photo-realistic scenes and offers various tools and third-party add-ons. UnrealCV is a tool supporting Unreal Engine 4, which is made to help with computer vision tasks.

Unreal Engine has some strengths compared to Unity. The assets and 3D-rendered graphics have an overall higher quality than in Unity. To create synthetic images, it is essential to be close to photo-realistic. However, as Vanherle *et* *al.* [15] noted, low-level features are not the most impactful thing for synthetic images to resemble. Rendering scenes in Unreal Engine is also faster than in Unity. As mentioned in **Section 4.1.1**, computing time is essential, as available computing power is limited. However, the difference isn't significant enough to influence the generation process heavily. On the other hand, the Unity community is larger than the Unreal Engine community, and more assets and support are available online. Having high-quality assets is a critical factor in creating the scene. Similar to Unity Perception, the Unreal Engine tool UnrealCV is available. Although the tools are made for similar tasks, such as object detection, UnrealCV only supports Unreal Engine 4, an outdated software version, and not Unreal Engine 5. Not only that, but the tool itself has not been updated since late 2017. All these aspects combined made Unity and Unity Perception and clear choice.

Leveraging GANs for Synthetic Image Creation

Generative adversarial networks (GANs) are a deep learning model that can generate realistic-looking images [66]. The model consists of two neural networks, a *generator* and a *discriminator*, used to learn features and patterns, enabling the network to create synthetic images that resemble real examples. The generator is responsible for learning patterns and generating examples, while the discriminator classifies the output, labelling it as real or as a generated image. By trying to fool the discriminator, the generator learns how to generate realistic-looking data. The back-and-forth between the two models continues until the generator can fool the discriminator with regularity.

One of the advantages of using GANs is data augmentation. Data augmentation is a proven technique to improve object detectors' performance [23]. Techniques for data augmentation to expand datasets with new instances are often primitive, with flipping, cropping, zooming or other simple transformations of datasets being usual. With GANs, it is possible to create new examples with more domain-specific features and variations than simple transformations [26][27][23]. Creating art, high-resolution versions of input data, and image-to-image translation [67] are some of the capabilities of GANs. Over the years, GANs have successfully created photo-realistic images of human faces, which are impossible to differentiate from real example data [68].

Even though GANs have a lot of potential for generating synthetic images for object detection applications, there are some disadvantages to the specific setting of this thesis. Firstly, developing GANs is a complex task, both regarding domain expertise, but also complexity regarding training [67]. Compared to a game engine like Unity, developing a GAN requires knowledge and expertise within deep learning to produce realistic, high-quality images. Even with high-powered computers available, the process is time-consuming and computationally demanding. With Unity, it is possible to quickly create photo-realistic images with less demand for power and less required expertise in the field.

One of the benefits of using Unity is the level of control it provides in creating a predictable and organized environment. When working on a thesis, positioning objects in relation to each other is crucial and cannot be left to chance. Unity makes it easy to adjust and modify the scene and objects. Adding new objects is a simple process, unlike in a GAN, where the system must first learn the features of the object before it can be inserted into the scene. Additionally, Unity allows for easy adjustments to lighting, camera angles, and object details. These features require extensive training and example data in a GAN.

4.6.2 Alternative Approaches for Object Detection Models

The object detection field has had massive progress over the years, and new emerging object detection models are released yearly. There are multiple options for the object detection model for sheep detection. This section will present alternative models and examine their advantages and drawbacks.

Similar to YOLO, Single-shot detector (SSD) is a single-stage object detection algorithm, meaning they perform predict object class labels and create bounding boxes in a single forward of the network. Despite both being single-stage detectors, some differences set them apart. One aspect that distinguishes them is the creation of bounding boxes. SSD uses anchor boxes, which are pre-defined bounding boxes of different sizes, to make predictions. These boxes are matched to the ground truths during training before adjusting box offsets and class probabilities to learn to detect objects. Contrary, YOLO does not use anchor boxes. Each image is divided into a grid, where each grid is responsible for predicting bounding boxes and probabilities.

SSD and YOLO are two algorithms that aim to balance speed and accuracy for object detection tasks requiring real-time, precise predictions. These algorithms are beneficial in autonomous driving, agriculture monitoring, or detecting sheep in UAV images. YOLOv7 is currently the latest object detection algorithm. It has been utilized in multiple earlier master theses, making it a suitable option for further studies and comparisons with previous work.

Kortylewski *et al.* [27] introduced the CompositionalNet algorithm focusing on occluded objects. Their results show significant increases in performance compared to standard CNNs at detecting occluded objects. Implementing this algorithm for finding sheep in UAV images can potentially improve the detection of occluded classes.

4.6.3 Weaknesses of Selected Approach

As mentioned in **Section 4.2.1**, Unity Percetion has some limitations. Given the limiting scope and time, some of these can be circumvented, but others are difficult to address.

A significant advantage of Unity is its publicly available free and paid assets. However, the data generation method is bound to use these assets, as it is not feasible to create enough custom assets due to limited time. Making them with the required quality to be useful for training would also be difficult. Some of the assets used, like the sheep and tree assets, are of questionable quality. The textures and looks have been acceptable for the experiments for this thesis, even though they have made the positioning of sheep challenging. An extensive experiment would require more precision to be optimal for occluded sheep training.

Even though Unity could run on available hardware, generating images has been slow, often susceptible to interruptions and crashes. Because of this, the number of new images generated of occluded sheep has been lower than initially desired. This can potentially influence the impact of occluded sheep images. It is clear that Unity Perception has a lot of potential for machine learning applications, but there are still glitches and errors that need to be addressed.

The selected object detection model, YOLOv7, also has some weaknesses. Muribø [10] reported that an earlier version of YOLO performed better on one superclass of sheep than on multiple sub-classes. YOLOv7 doesn't support super and sub-classes, so these configurations could not be tested, as it potentially would influence the models' ability to detect sheep.

Chapter 5

Experiment Structure

The following chapter explains how the experiment for this thesis is structured and how the experiments relate to the research questions. The research questions will be presented, along with the approach for training and testing.

5.1 Research Questions

RQ1 *How does YOLOv7 perform with the dataset consisting of real images?*

From previous work, the latest object detection model used on the sheep dataset is YOLOv5. According to [49], YOLOv7 is up to 120% better than YOLOv5 on the MS COCO dataset. Given the dramatic increase in performance, the hypothesis is that YOLOv7 should perform significantly better than its predecessor. As the data used for training will be exclusively real data, the results of the YOLOv7 model will be the *baseline* for further experiments and comparisons.

RQ2: *How does training on mixed synthetic and real data affect the detection performance of occluded sheep?*

The newly generated synthetic data will be mixed with real image data to train two YOLOv7 models, evaluating their ability to detect occluded sheep. These models will be compared to the baseline model from the previous research question. The aim is to ascertain how the synthetic data affects the performance. The hypothesis is that adding synthetic data should improve the performance of the occluded classes in the dataset.

RQ3: *How does fine-tuning compare to mixed training performance for predicting occluded sheep?*

Previous research has studied how mix-training with synthetic data affects the models' ability to detect sheep [13]. According to [15], training on synthetic data before fine-tuning outperforms mix-training in accuracy, requiring less training time. The results from fine-tuning will be compared to the results of mixed training obtained from the prior research question, considering their accuracy. The hypothesis is that fine-tuning should produce better results than mixed training results.

5.2 Dataset Splits

The dataset described in **Section 4.4** will be used for training, validation and testing. **Table 5.1**, **Table 5.2**, and **Table 5.3** give an overview of the number of class instances in each of the subsets.

The dataset of real images is split so that 70 % of images are in training and 15 % in each of the validation and testing subsets. The dataset of synthetic images is split so that 90 % of the images are in training and 10 % are in validation. All classes must be represented to make the datasets as useful as possible for training and testing. Still, because of the highly imbalanced nature of the real dataset, it is challenging to provide enough unique examples of underrepresented classes for training, validation and testing. It will most likely impact both training and testing.

Each subset is also split into real and synthetic data. The *Real* data is all 2,125 images from the real dataset, and the *Synthetic* is the combination of synthetic data generated for Østtveit [13] and this thesis. Synthetic data is primarily used for training, but the validation of models trained on exclusively synthetic data is only validated on synthetic data. The other models use real image data for validation, and all models use exclusively real data for testing.

Training			
Class	*Real*	*Synthetic*	*Total*
Black sheep	1,638	2,190	3,828
Brown sheep	744	2,860	3,604
Grey sheep	2,427	1,488	3,915
White sheep	8,585	782	9,367
Black occluded sheep	26	1,487	1,513
Brown occluded sheep	6	1,415	1,421
Grey occluded sheep	87	1,524	1,611
White occluded sheep	627	2,925	3,552
Total	14,140	14,671	28,811

Table 5.1: Training data distribution across classes

Validation			
Class	*Real*	*Synthetic*	*Total*
Black sheep	360	211	571
Brown sheep	165	322	487
Grey sheep	546	153	699
White sheep	1,910	69	1,979
Black occluded sheep	6	185	191
Brown occluded sheep	2	173	175
Grey occluded sheep	31	175	206
White occluded sheep	139	320	459
Total	3,159	1,608	4,767

Table 5.2: Validation data distribution across classes

Testing	
Class	*Real*
Black sheep	364
Brown sheep	162
Grey sheep	542
White sheep	1,885
Black occluded sheep	8
Brown occluded sheep	2
Grey occluded sheep	31
White occluded sheep	138
Total	3,132

Table 5.3: Testing data distribution across classes

5.3 Training

Pytorch was used to define the YOLOv7 architecture for training the sheep object detector. It's a framework used for machine-learning applications and can be used to build deep-learning models. The framework is open source with lots of resources available. It enables tensor computation on Graphical Processing Units and is built with Python.

All models are trained on the IDUN cluster at NTNU. The cluster consists of many nodes with variations in CPUs, number and types of GPUs, and memory sizes. The NVIDIA A100 40GB was used for training.

Table 5.4 gives an overview of the models trained to test the research questions for this thesis. All models trained are the standard YOLOv7 model with default hyperparameters, except for a lower learning rate. All real images are tiled

ID	*Model*	*Training*	*Dataset*	*Class Type*	*Epochs*
1A	YOLOv7	Scratch	Real	Multi class	500
1B	YOLOv7	Scratch	Real	Single class	500
1C	YOLOv7	Pre-trained	Real	Multi class	500
1D	YOLOv7	Pre-trained	Real	Single class	500
2A	YOLOv7	Scratch	Mixed	Multi class	500
2B	YOLOv7	Scratch	Mixed	Single class	500
3A	YOLOv7	Scratch	Synthetic	Multi class	200
	YOLOv7	Fine-tuning	Real	Multi class	500
3B	YOLOv7	Scratch	Synthetic	Single class	200
	YOLOv7	Fine-tuning	Real	Single class	500

Table 5.4: Configurations of models used for training and testing

to 640px by 640px, the same size as the synthetic images.

Preliminary testing of training parameters was conducted to find optimal values for the number of epochs and learning rate. Most of the models are trained for 500 epochs, as it proved to be a good amount for yielding valuable results. In combination with YOLOv7s built-in *Early Stopping* feature, which stops training after a defined number of epochs without any improvement, the training process can avoid using unnecessary resources. The training from scratch of model *3A* and *3B* are run for 200 epochs, which preliminary testing suggested to be sufficient and avoided overfitting. After experimentation with lowering the learning rate, it was lowered from the default of 0.01 to 0.001 for all models because of early overfitting in initial testing. The primary focus for training is to evaluate the performance of models trained on different combinations of the datasets described in Section 5.2. The *Real* and *Synthetic* using the respective dataset, and *Mixed* utilising the combination of both for training. For mixed, all the synthetic images are used in training. Each model is also trained on multiple and single classes. For the use case of the UAV, having a single class of sheep would be beneficial. For the thesis, it is advantageous to study the performance of individual classes so that they can be inspected closer and examined further down the line.

Model *1A-1D* are the models related to RQ1, which tests how YOLOv7 is performing on the real dataset. Training from scratch and fin-tuning on pre-trained weights on the MS COCO dataset for both single and multi-class is tested. The best-performing model will be the *baseline* for comparison with the other models for the rest of the experiment. Model *2A* and *2B* are tested for RQ2. The model will be trained on combining the real and synthetic datasets, both single and multi-class, to see how mixed training performs with synthetic data. Finally, model *3A* and *3B* related to RQ3 will test how fine-tuning is compared to mixed training. The training of these models consists of two phases. First, the models will be trained from scratch on the synthetic dataset before it is fine-tuned on the real images. This will be done for both multi and single-class models. The results from both pre-training and fine-tuning will be examined during training, but only the fine-tuned model will be tested further.

5.4 Testing

When all configurations are done training, the models with the best-performing weights through training will be selected for testing on the test data split. The best-performing model in YOLOv7 is selected using a *fitness function*. The fitness function outputs the weighted average of precision, recall, mAP@0.5 and mAP@0.5:0.95 from the validation set to determine the best-performing model. Using the best model compared to the latest model is done to avoid using an overfit model, which probably will perform worse on the unseen data in the test data split. All tests are run on the IDUN cluster with the NVIDIA P100 node.

For evaluating the sheep detector models, different evaluation metrics will be used. Some are more relevant for the application of finding sheep, and some for the focus of this thesis specifically. When looking at the multi-class models, it is important to note that even though there are different classes with different colours and levels of occlusion, all classes are still sheep. Therefore, a false positive prediction could be a sheep of another colour or a sheep hidden behind a branch. This will influence the precision and recall score of the model. However, it will still be a positive prediction when the goal is finding sheep, even though it belongs to another class. This could be a result of the labelling of sheep and where the limits are drawn in the transition from one colour to another, for example, if a grey sheep is predicted to be a white sheep. Nevertheless, keeping the number of false negatives in mind is also important, as too many positive predictions of background objects would drastically decrease the value of the object detector. Because of how the sheep are classified, looking at the percentages of sheep being found can be valuable. It can be found through a multi-class confusion matrix.

For the single-class models, precision and recall are more valuable. A false negative in a single-class model could only be a missed ground truth because no other classes are being predicted. Reporting mAP@0.5 and mAP@0.5:0.95 will also be of value for combining the precision and recall metrics and evaluating bounding box precision.

Chapter 6

Results

The following chapter will present the results of the experiments conducted for this thesis. First, results from training all models in **Table 5.4** will be examined. Furthermore, the results from testing these models on the test dataset will be presented, which will be used to compare the performance across configurations.

6.1 Results from Training

The training of the models where all performed on two NVIDIA A100 40GB GPUs. Running time varied across datasets but was fairly equal between single and multi-class models. The baseline models trained from scratch, *1A* and *1B*, finished in approximately 11 hours, while the baseline pre-trained models, *1C* and *1D*, in 4 hours. The training from scratch on synthetic data for models *3A* and *3B* took about 4 hours and 30 minutes, while the fine-tuning itself took about 11 hours and 30 minutes. The outlier was the mixed dataset, which finished in 19 hours and 20 minutes for the multi-class model and 22 hours and 37 minutes for the single-class model. It is not surprising that the training took longer than for the other models, as the combination of the two datasets is larger than any one of them. However, the difference between the two models concerning the number of classes is more surprising, as the models have the same hyperparameters and are trained on the same hardware. These differences might result from randomness during training, like the randomised initial weights.

Figure 6.3 and **Figure 6.2** show the mAP@0.5 for the trained models. The metric is calculated from the validations set after each epoch is finished. 500 epochs are clearly enough as the performance of all models is starting to plateau around 200-300 epochs. Looking at **Figure 6.4**, objectness loss is increasing as the models have trained for 200-300 epochs, indicating they are starting to overfit. These results suggest that the models have been trained for long enough.

An outlier in **Figure 6.3**, **Figure 6.2** and **Figure 6.4**, is the pink graph, representing the pre-training on synthetic data of models 3A and 3B. It is trained for fewer epochs to avoid overfitting, which can occur if the synthetic training data isn't varied, large, and representative of the real data [57]. Even though the synthetic data is created to be representative and varied, its accuracy to the real images isn't exact, and the number of generated images is much lower than suggested in Vanherle *et al.* [15] experiments, making it prone to overfitting. However, **Figure 6.4** shows the object loss has only slightly increased in the last half of epochs. Because of the small increase, letting the training continue longer might be favourable, but it would have to be explored further. It should also be noted that the same models perform very well on the mAP@0.5 score, especially for the multi class model, as seen in **Figure 6.2**. This drastic increase results from the models being validated on synthetic images alone, and the dataset is more balanced than the real images. There are more instances of the dark-wooled and occluded classes, increasing the AP scores for all classes, resulting in a high mAP score.

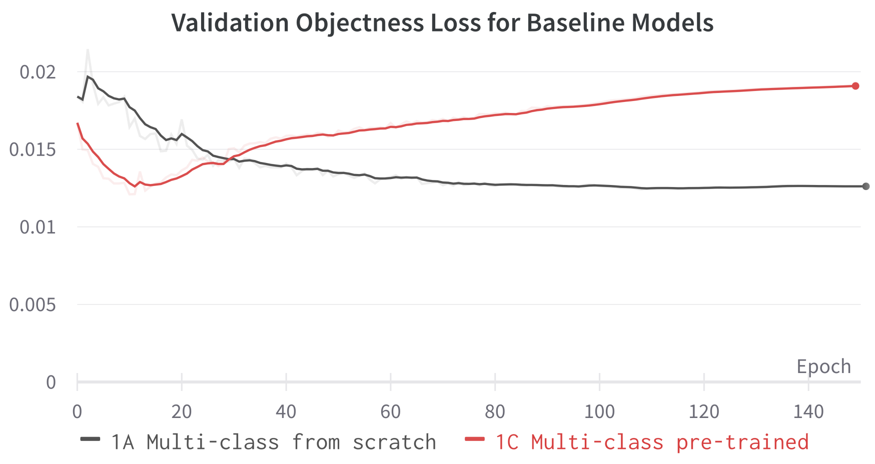


Figure 6.1: Comparing validation objectness loss for baseline models pre-trained and trained from scratch

The pre-trained model 1A, trained on the MS COCO dataset, performed worse during training than model 1C, trained from scratch. Looking at **Figure 6.1**, the validation objectness loss for model 1C has a quick dip before it starts to increase as early as before epoch 20. It continuously increases for the rest of the training epochs. It might seem that model 1C gets a low objectness loss quicker than model 1A before it starts to overfit. This could be looked at as advantageous, requiring less training time. However, in the early epochs, both models had relatively low mAP@0.5 scores compared to what they achieved in later epochs. The same tendencies showed for the single-class models 1B and 1D. Because of these tendencies during training, the models trained from scratch were selected as the baseline model for comparison for the rest of the experiments.

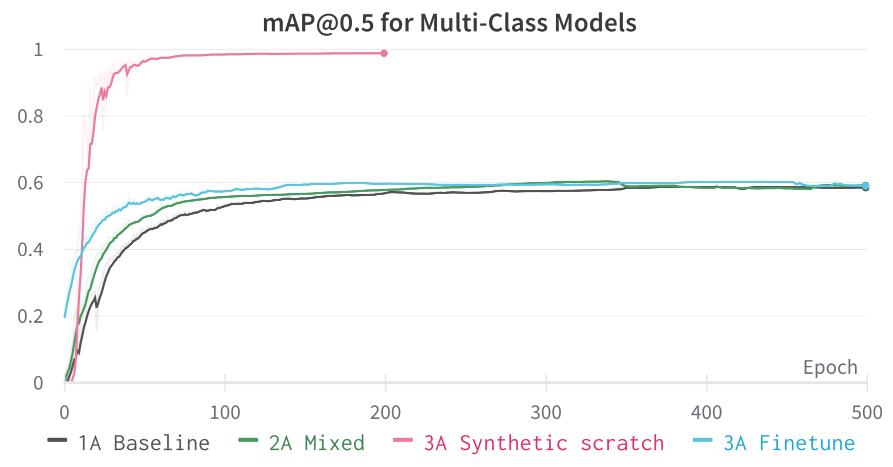


Figure 6.2: mAP@0.5 for all multi-class models

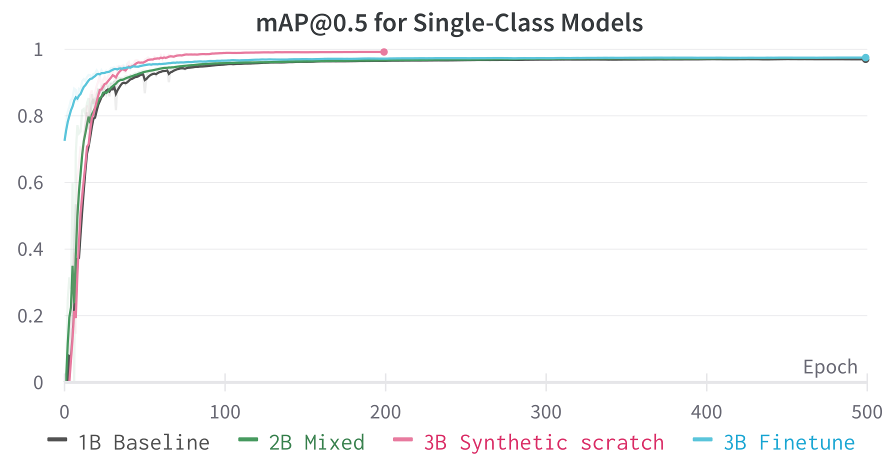
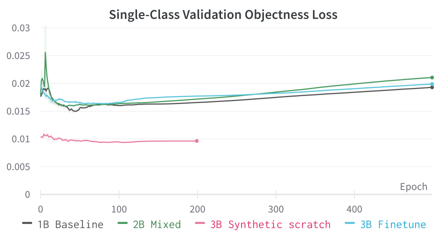
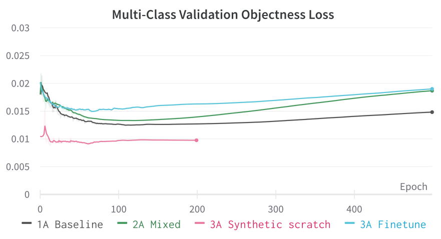


Figure 6.3: mAP@0.5 for all single class models



(a) Single class



(b) Multi class

Figure 6.4: Validation objectness loss for single and multi-class models

6.2 Results from Testing

Examining Evaluation Metrics

Table 6.2, **Table 6.3**, **Table 6.4** and **Table 6.5** shows the results from testing all sheep detection models on the test data set, with the highest values highlighted. The optimal confidence thresholds found from the F1 curve for each model are used to calculate the metrics for all models, which can be seen in **Table 6.1**.

Optimal Confidence Thresholds		
	A Multi	*B Single*
1 Baseline	0.602	0.798
2 Mixed	0.639	0.790
3 Fine-tune	0.631	0.787

Table 6.1: Optimal confidence thresholds used for testing

Each multi-class model has its respective values for each class in the tables. The single-class *all* value and the multi-class *all* values are differentiated by the fact that the multi-class is the mean of all other classes detected, meaning the weaker performing classes pull the mean down. In contrast, the single class is trained and tested as one class, meaning the weak-performing classes don't impact the mean extensively because there are so few examples.

As evident, most scores are pretty similar. However, there are some outlier values and classes. Most notable are the *brown occluded* class. All the different models struggle with finding this class, with all having 0 precision and recall at the optimal confidence threshold. Another apparent value is model 1A's precision score of 1 and recall score of 0.141 for the *black occluded* class. For this class, the other models have noticeably lower precision and higher recall. For both cases with the brown and black occluded classes, it is important to note the number of instances in the test dataset, which is 2 and 8, respectively. There are few instances, meaning each prediction greatly impacts the metrics. Also note the relationship between precision and recall, resulting in the baseline model prediction of all black sheep correct but simultaneously not finding many of them. Models 2A and 3A, however, are finding more, but with less precision. Neither of the models could find any of the two brown occluded sheep in the dataset. These test set limitations should be considered when evaluating the results.

Even though the results are similar for many classes and models, there are some trends. For the mAP scores, the models trained on synthetic data consistently outperforms the baseline, if ever so slightly. The only class where the baseline has better mAP scores are the *brown sheep* class. Of the models trained on synthetic data, the mixed models perform the best for multi-class, achieving a higher mAP@0.5 and mAP@0.5:0.95 than single. For the single class, 3B has the highest mAP@0.5 and 2A for the mAP@0.5:0.95 scores.

Precision			
Multi Class	*1A Baseline*	*2A Mixed*	*3A Finetune*
all	**0.681**	0.676	0.660
black sheep	0.841	**0.877**	0.851
brown sheep	**0.733**	0.708	0.719
grey sheep	0.811	**0.847**	0.42
white sheep	0.866	**0.917**	0.909
black occluded sheep	**1.000**	0.897	0.686
brown occluded sheep	0.000	0.000	0.000
grey occluded sheep	0.693	0.607	**0.750**
white occluded sheep	0.500	**0.552**	0.525
Single Class	*1B Baseline*	*2B Mixed*	*3B Finetune*
all	**0.940**	0.937	0.937

Table 6.2: Precision scores from testing

Recall			
Multi Class	*1A Baseline*	*2A Mixed*	*3A Finetune*
all	0.606	0.631	**0.630**
black sheep	**0.890**	0.871	0.887
brown sheep	**0.849**	0.839	0.838
grey sheep	**0.913**	0.910	0.904
white sheep	**0.962**	0.961	0.959
black occluded sheep	0.141	**0.375**	**0.375**
brown occluded sheep	0.000	0.000	0.000
grey occluded sheep	0.452	0.452	**0.484**
white occluded sheep	**0.645**	0.638	0.594
Single Class	*1B Baseline*	*2B Mixed*	*3B Finetune*
all	0.929	0.947	**0.950**

Table 6.3: Recall scores from testing

mAP@0.5			
Multi Class	*1A Baseline*	*2A Mixed*	*3A Finetune*
all	0.626	**0.653**	0.637
black sheep	0.883	**0.896**	0.880
brown sheep	**0.799**	0.793	0.789
grey sheep	0.914	**0.920**	0.918
white sheep	0.968	**0.971**	0.968
black occluded sheep	0.430	**0.562**	0.489
brown occluded sheep	0.002	0.000	**0.005**
grey occluded sheep	0.481	0.488	**0.510**
white occluded sheep	0.530	**0.592**	0.536
Single Class	*1B Baseline*	*2B Mixed*	*3B Finetune*
all	0.959	0.963	**0.966**

Table 6.4: mAP@0.5 scores from testing

mAP@0.5:0.95			
Multi Class	*1A Baseline*	*2A Mixed*	*3A Finetune*
all	0.395	**0.421**	0.408
black sheep	0.594	**0.604**	0.595
brown sheep	0.518	**0.538**	0.532
grey sheep	0.618	0.632	**0.640**
white sheep	0.728	**0.739**	0.737
black occluded sheep	0.187	0.284	0.228
brown occluded sheep	**0.001**	0.000	0.000
grey occluded sheep	0.209	0.210	**0.228**
white occluded sheep	0.306	**0.358**	0.305
Single Class	*1B Baseline*	*2B Mixed*	*3B Finetune*
all	0.691	**0.697**	0.696

Table 6.5: mAP@0.5:0.95 scores from testing

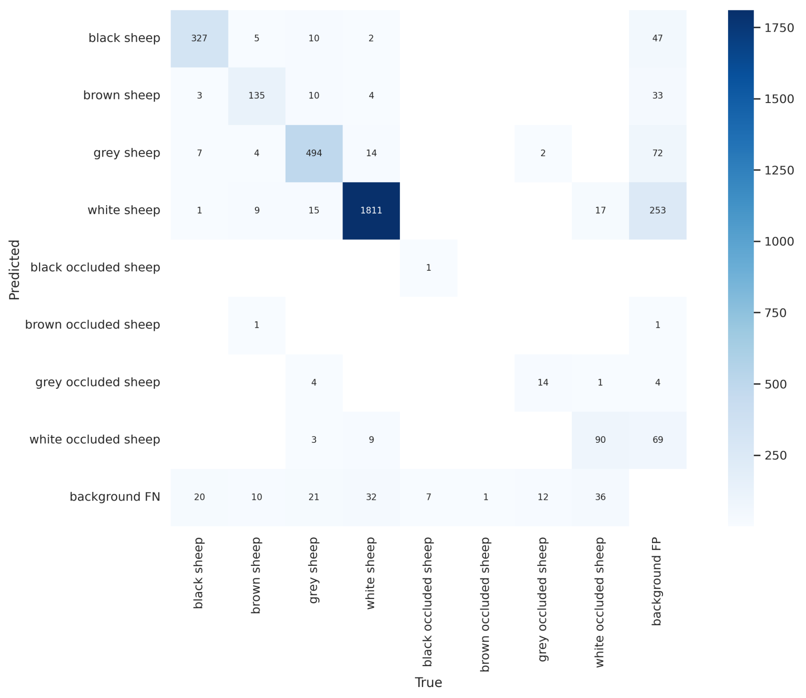
Examining Confusion Matrices

Figure 6.5, **Figure 6.6** and **Figure 6.7** shows the confusion matrices for the multi-class models *1A*, *2A* and *3A*. These matrices are calculated using the optimal confidence threshold mentioned earlier. Examining the matrices shows which classes are wrongly predicted during the prediction process. A false positive between classes is not crucial for finding sheep but gives insight into the model's performance.

Looking at the matrices, it is clear that the *white sheep* class is the most abundant in the dataset. There are many more instances of the class, but all models also detect a high degree of them. The high detection rate is no surprise, given the number of instances in the training set. Many class predictions are falsely predicted as other sheep classes, especially those of the non-occluded classes. Some non-occluded classes are also being predicted as the occluded class with the same colour. By adding the number of true and false positives, the exact number of positive predictions can be found, giving a more accurate impression of the multi-class models' performance.

The detection of occluded sheep has some varying results. The mixed and fine-tuned models have found more black-occluded sheep than the baseline model. There are only eight instances in the test set, but three correct predictions are still an improvement from one correct prediction. As with the brown occluded class, non of the models managed to get a correct prediction. However, both the models trained on synthetic data managed to predict brown occluded sheep as another class of sheep. The mixed model 2A had two false positives, which were classified as white sheep, and the fine-tuned model 3A predicted them to be of the white and white occluded classes.

Another result which should be noted is the number of false positives in the multi-class models. **Table 6.1** show the optimal thresholds for all models. Most notable is the difference between the multi-class models, especially the lower threshold for the baseline model 1A. This can also be seen in the confusion matrices, where it is evident that model 1A exhibits a higher rate of false positives compared to the other two multi-class models, despite almost predicting the same number of sheep correctly. The white sheep classes seem particularly affected, with 253 false positive white sheep instances in model 1A, as opposed to 148 and 135 in the mixed and fine-tuned models. Other non-occluded classes are also seeing a decrease in false positives. It is more difficult to determine for the occluded classes, as the number of predictions is so low. However, the number of white-occluded false positives decreases on both models trained on synthetic data. The only increase is the number of false positives for the grey-occluded sheep class for the mixed model.



	black sheep	brown sheep	grey sheep	white sheep	black occluded sheep	brown occluded sheep	grey occluded sheep	white occluded sheep	background FP
black sheep	327	5	10	2					47
brown sheep	3	135	10	4					33
grey sheep	7	4	494	14			2		72
white sheep	1	9	15	1811				17	253
black occluded sheep					1				
brown occluded sheep		1							1
grey occluded sheep			4				14	1	4
white occluded sheep			3	9				90	69
background FN	20	10	21	32	7	1	12	36	

Figure 6.5: Confidence matrix for model 1A Baseline

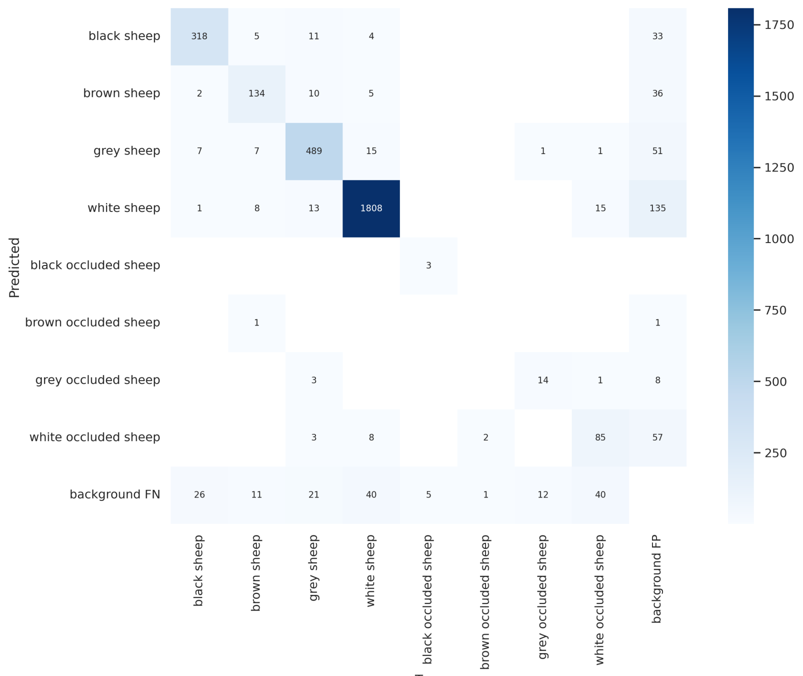


Figure 6.6: Confidence matrix for model 2A Mixed

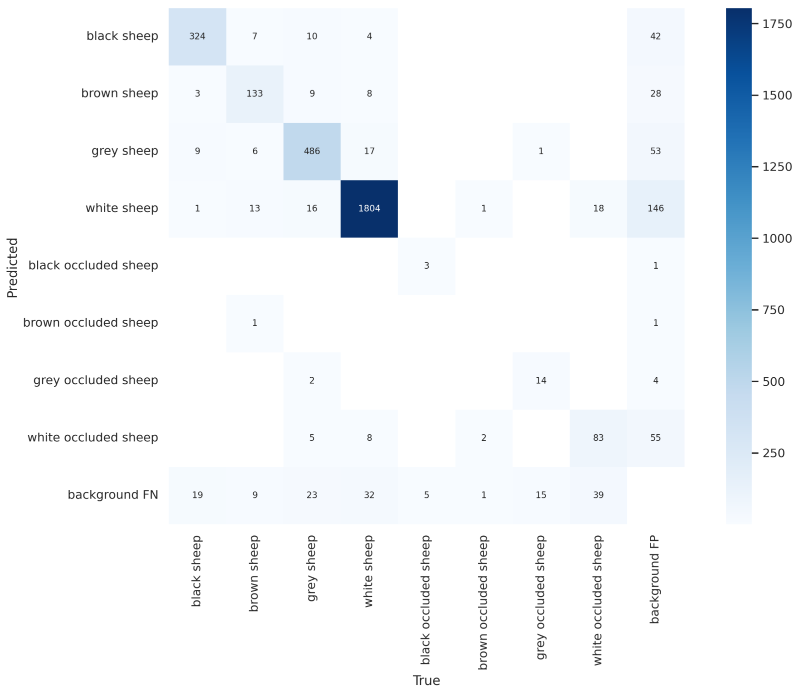


Figure 6.7: Confidence matrix for model 3A Fine-tuned

Chapter 7

Discussion

The following chapter will examine the results and discuss them with regard to the research questions. Lastly, different sources of errors from training and testing which can influence the results will be considered.

7.1 Research Question 1

RQ1 *How does YOLOv7 perform with the dataset consisting of real images?*

To assess *RQ1*, the baseline YOLOv7 models performance on the existing dataset will be compared to the YOLOv5 baseline model used in last year's study by Østtveit [13]. The single-class configurations were used for this comparison since the YOLOv5 metrics were reported as a single-class model. This offers a more precise evaluation of how the models perform for the application of finding sheep but doesn't make it possible to compare the performance across multiple classes. Both models were trained on an identical dataset and employed the default hyperparameters other than the learning rate for their respective architectures. **Table 7.1** showcases the primary metrics for both models.

Architecture	*Precision*	*Recall*	*mAP@0.5*	*mAP@0.5:0.95*
YOLOv7	0.940	0.929	0.959	0.691
YOLOv5	**0.947**	**0.946**	**0.962**	**0.747**

Table 7.1: Comparison of YOLOv7 and YOLOv5 performance

The YOLOv5 models from the experiments by Østtveit [13] outperform the YOLOv7 model on all evaluated metrics. While the differences are not enormous, the discrepancies are intriguing since the older YOLOv5 model surpassed the more recent SOTA YOLOv7 model. The most significant difference was observed in the mAP@0.5:0.95 scores, indicating that the bounding boxes predicted by YOLOv5 are more precise than those predicted by YOLOv7.

Various factors stemming from the architecture's differences could influence the models' performance, like how each model handles different data types and image complexity. Each version of YOLO introduces changes and refinements in the network architecture, loss function, or training process, which may have varying implications for different datasets.

Additionally, the hyperparameters used for both models might benefit one architecture over the other. Although the models were trained with their respective default hyperparameters, except for a lower learning rate on the YOLOv7 models, other settings might not have been optimal for this particular dataset, thus affecting the performance of YOLOv7 more negatively than YOLOv5.

Furthermore, it is also essential to consider that many variables can influence the performance of machine learning models on a specific task. The particular characteristics of the dataset, including the complexity of images, diversity in sheep classes, and the level of occlusion, might be better suited to the YOLOv5 model.

However, further research and experimentation would be required to determine the causes for the difference in performance, involving adjustments in hyperparameters, changes in training, analysis of performance across classes, and how the models perform on varying degrees of image complexity.

7.2 Research Question 2

RQ2: *How does training on mixed synthetic and real data affect the detection performance of occluded sheep?*

To assess *RQ2*, the performance of models 2A and 2B will be examined. The two models' results will be compared to the baseline models 1A and 1B to evaluate the performance of detecting occluded sheep with mixed data.

Model 2A, trained on multiple classes, generally outperformed the baseline model 1A regarding precision and recall. One notable difference between these models is their respective confidence thresholds. Model 1A has a lower optimal confidence threshold than the mixed model, explaining its lower precision due to a higher number of false positives. However, this should also result in a higher recall, as it requires less confidence to label an object as a member of the sheep class. Despite these differences, the recall scores of both models tend to be pretty similar across most classes, usually varying by just a few hundredths. This suggests that the mixed models can make high-precision predictions while maintaining similar recall scores. The benefit of using mixed models is further evident in the mAP@0.5 scores from **Table 6.4**, where they outperform the baseline model for all classes except the brown sheep class.

The mixed models show promising performance when dealing with occluded classes. For instance, the black occluded class witnessed a significant increase in mAP@0.5 performance from *0.430* to *0.562*, while the white occluded class experienced an increase from *0.530* to *0.592*. However, it's crucial to note that the test dataset only includes eight black occluded instances, thus suggesting a need for a more comprehensive dataset for reliable evaluation of the real impact of synthetic images. Despite having more instances in the training set to learn from, the benefits of recognizing these features may not be fully realized. The occluded white sheep implies that incorporating the occluded sheep dataset could be beneficial. Still, for the black and brown occluded sheep, there might not be enough test images to see the performance improvement, making it challenging to make a definitive conclusion.

Regarding the single class models, 1B and 2B, the mixed model, 2B showcased a significant improvement in recall with *0.947* compared to *0.929* for the baseline model 1B while maintaining a comparable precision score, from *0.937* compared to *0.940*. In single-class models, although we cannot identify which sheep class the model has difficulties recognizing, the increase in recall, combined with similar precision, suggests the model's enhanced ability to detect more sheep, albeit without perfect classification. For the drone's practical use, this is not problematic. It's important to remember that the primary goal is to ensure all sheep are detected, even if the exact classification may not always be accurate. Therefore, the precision-recall tradeoff can be considered suitable in this use case.

Overall the results indicate some improvements in occluded sheep detection using mixed data. Experimenting with ratios between real and synthetic data could improve the results. Results from Vanherle *et al.* [15] suggested a ratio of 5:1 between synthetic and real images. Testing with different ratios could be beneficial for the application of sheep detection. But still, the real value of adding synthetic occluded sheep images to the training data will need further verification through testing with a larger, more balanced dataset.

To improve the results further, changes in the approach to generating synthetic data could be explored. Specifically, reconsidering the synthetic data used to enhance occluded sheep detection may be beneficial. A more diverse set of generated scenes could provide the model with a wider variety of contexts for learning. This could involve enhancing the landscape and sheep's visual attributes in these synthetic images. Moreover, the image generation process could be focused more on underrepresented sheep classes in the dataset. While it is crucial to generate images covering all sheep classes to ensure comprehensive feature learning, prioritizing the creation of dark-wooled occluded sheep would most likely boost the model's performance. In this way, efforts would be more strategically targeted, enabling a more comprehensive learning process for the model and, potentially, better results for classes with the worst performance.

7.3 Research Question 3

RQ3: *How does fine-tuning compare to mixed training performance for predicting occluded sheep?*

Analyzing the performance and differences between mixed and fine-tuned training focuses on models 2 and 3. As an overall trend, the mixed model surpasses the fine-tuned model in most performance metrics. When considering the mAP@0.5 score, the fine-tuned model lags in all classes except for brown and grey occluded. For the non-occluded classes, performance is lower, albeit not so much. The difference is more significant with the occluded classes, where the black and white occluded sheep scored 12.9% and 9.4% lower than the mixed model, respectively.

While these deviations aren't vast and originate from a test set with a limited size for the relevant classes, they might not hint at a broader trend outside these experiments. Contrary to these findings, Vanherle *et al.* [15], Seib *et al.* [23], and Nowruzi *et al.* [69] reported superior results and performance with fine-tuning instead of training on a mixed dataset. However, from the sheep data experiments, the fine-tuned model performed worse. A key constraint for the fine-tuned models 3A and 3B is the amount of training data available. A large and diverse synthetic dataset is crucial for optimal fine-tuning [55], making it a more viable alternative to exclusively training on real data. Unfortunately, due to time and hardware limitations, generating the necessary quantity of data was unfeasible. Other causes for the diverging results could be related to the pre-training with exclusively synthetic data. From the results from training, described in **Section 6.1**, the pre-training could have been prolonged. It is not certain, however, that it would lead to increased performance. Another option could be experimenting with freezing the shallow layers after the model is pre-trained. Seib *et al.* [23] found the technique to give promising results, while in contrast, Vanherle *et al.* [15] got better results without freezing the layers and rather let them continue training for longer, concluded that it should be considered on a per-problem basis. Further exploration of the technique could give better results from the fine-tuning of the models. Ratios of synthetic and real images could also be explored, as a 1:1 ratio got the best results in the same study. The ratios between real and synthetic data across classes varied immensely and could be optimised for better results. However, it is still limited by the number of real images of certain classes.

Enhancing the efficiency and capabilities of the synthetic image generator could be a feasible option for improving the models' performance. Specifically, runtime and image variation have room for improvement. A more effective generator that produces a wider variety of sheep images is a viable option for further bettering pre-training for the fine-tuned model.

7.4 Implications and Limitations

7.4.1 Practical Implications on UAV Roundup

With the results from the experiments, camera-equipped UAVs could be a viable option for sheep roundup. However, such a system for detecting sheep would need to be developed further beyond the scope of this thesis to be a real solution for the commercial market. For the detection of sheep, especially occluded sheep, to be advanced further, it would require more quality data. Naturally, in machine learning applications, the performance and results from a model are heavily influenced by available data. As mentioned, training and testing the object detection models require more data, as the lack of real examples of occluded sheep of all colours is a barrier to further development. Synthetic data could help performance, but real data will always be needed. Innovations in sheep detection performance will require more real image data and should be a focus for future work on the subject.

7.4.2 Limitations and Sources of Error

Label Quality

Labelling image data is a difficult task which requires precision and patience. The dataset consists of thousands of images and thousands of instances of sheep, which have been labelled by hand. Even though the sheep dataset is relatively small, human errors will occur during labelling. Occluded or dark-wooled sheep in shadows, which may not be easily visible at first glance, can be missed and left unlabeled, resulting in the model classifying them as false positives. Labelling a sheep to the correct class can be challenging at times. Some sheep exhibit multicoloured fur; the lightest grey sheep may appear white, and the darkest brown sheep may appear black. Distinguishing between these variations and many others can be difficult. Furthermore, factors such as occlusion, shadows, and lighting conditions introduce additional challenges to the labelling process, potentially causing the model to misclassify them. In addition, there may be objects in the images resembling sheep that could be mistakenly labelled as sheep, leading to the model being trained incorrectly and resulting in increased detection of false positives by the model. Even if the labelling is correct, the quality of the bounding box annotations may fall short. Imprecision in the bounding boxes, such as not capturing the entire sheep, including space around the sheep, or failing to label occluded sheep as fully visible, can affect the training process. These mistakes can result in a poorly trained model with decreased performance.

Mesh Quality in Unity

The mesh quality utilized for the trees in Unity is not perfect. While it approximates the trees' shape, adding textures does not provide a precise 1-to-1 representation. This inconsistency can lead the script to perceive the sheep as occluded when it is actually fully visible but standing near the tree. It can also perceive the sheep as occluded when the sheep is entirely invisible under the tree. As these labels are automatically generated, they don't require any visible sheep, making it impossible for the model to learn any meaningful feature from the image.

Background Images

A background image is an image that does not contain any labelled objects. The sheep dataset does not contain any background images. To reduce the number of false positives detected, having between 0 to 10% background images is recommended.

Random Weights Initialziation

YOLOv7 uses random starting weights during training. Utilizing this, each model trained can learn from a unique starting point, resulting in each model potentially learning and performing differently.

Quality of Datasplit

Some images in the dataset were captured in quick succession. This is not considered when dividing the dataset into training, validation, and testing sets. As a result, there is a possibility that similar images may end up in different sets. This can lead to a potential issue of overfitting during training.

Inconsistency in Confusion Matrices

Due to a bug in YOLOv7's code, inconsistencies have been observed in the confusion matrices. This bug leads to a few instances of sheep not being accounted for while a few extra instances that do not exist are mistakenly included. However, it is essential to note that the impact of these inconsistencies is relatively small compared to the total number of instances in the dataset.

Chapter 8

Conclusion

The following chapter will summarise the conclusion for each research question to evaluate if the goal posed for the thesis is accomplished. Suggestions for future work will also be presented.

8.1 Conclusion

RQ1 *How does YOLOv7 perform with the dataset consisting of real images?*

YOLOv7 performs worse than the older YOLOv5 architecture. The hypothesis was that the newer algorithm would perform better than the old one. However, our results suggest that the older architecture is better suited for detecting sheep in UAV images when it comes to dataset size, parameters and size of the model.

RQ2: *How does training on mixed synthetic and real data affect the detection performance of occluded sheep?*

Some indications suggest that training on mixed data of real and synthetic occluded sheep enhances the performance of the models. However, these are only indications from a small dataset. It must be tested on a dataset with more instances of all occluded classes to conclude anything with certainty.

RQ3: *How does fine-tuning compare to mixed training performance for predicting occluded sheep?*

Training the models from scratch on synthetic data before fine-tuning them on real data shows some potential. However, the results from these experiments do not have all the necessary data for training to display its full potential. Generating more synthetic images would probably produce better results.

Goal: *Improve detection of partially occluded sheep by training YOLOv7 on real and synthetic data generated with Unity Perception*

Based on the results of the research questions in this thesis, using Unity Perception to produce synthetic images is a viable approach to supply the sheep dataset with instances of occluded sheep. Models using the updated dataset show some indications of enhanced performance but need to be tested on a dataset with more examples of real occluded sheep to determine its effect with certainty.

8.2 Future Work

The use of synthetic images to enhance the performance of camera-equipped UAVs for sheep roundup has shown potential, but there are more options for future work. These suggestions are based on work that could not be included due to limited time, work that could be improved upon further, or options discovered while working on the thesis.

Utilze GANs for Synthetic Data Generation

As mentioned in **Section 4.6.1**, the possibility of utilising GANs has much potential for generating synthetic data for training object detection models [27][26]. The most imposing threat to utilising GANs is the lack of real data in the existing dataset, as the network requires a lot of data for learning and generating good examples for training a model. Because of this, it is advised to capture more images for potential future work on the application of camera-equipped UAVs in finding sheep. Without further expansion of the dataset, increasing the value of the object detector models might be challenging. With an expanded dataset, GANs can be explored to expand the synthetic dataset further.

Extend Use of Unity for Synthetic Data Generation

As shown in this thesis and other related work [13][24], using Unity and Unity Perception data is a viable option for generating synthetic data for training object detection models. This work can be elaborated on further. As shown by Oksuz *et al.* [14], Seib *et al.* [23], and Wang *et al.* [26], using synthetic data for providing underrepresented classes in imbalanced datasets can be experimented on further. Focusing more on the black and brown sheep, both occluded and non-occluded, can make the dataset more balanced. Devoting more time to generate synthetic data in Unity with these classes will potentially be of value to the application of finding sheep. The landscape created in Unity might also be further developed by creating a more realistic-looking landscape with more variation in the appearance of the occluded sheep. The algorithm generating occluded sheep could also be improved, as the sheep placement is confined to a smaller area in the image, reducing the potential variation and randomisation. Creating more varied instances of occluded sheep might potentially increase the value of the synthetic images.

Optimise UAVs Flight Altitude

Using a generated scene in Unity, optimising the flight height for the UAV is possible. Exploring sheep's optimal height and size in the images and optimising the area covered by the UAV's camera can be of value. It could optimise the time spent flying the drone and improve the object detector's accuracy.

Explore Alternative Object Detector Algorithms

As mentioned in **Section 2.5**, CompositionalNet is introduced to detect better and classify occluded objects and found through experiments that it does detect and classifies vehicles signify better than standard DCNNs. It would be interesting to see if this method could be successfully implemented to detect and classify occluded sheep.

During the writing of this thesis, two newer versions of YOLO have been released. YOLOv6 v3.0 and YOLOv8. Both versions claim to achieve a higher and faster mAP score on the COCO dataset than YOLOv7. This makes the new models also likely to detect sheep and occluded sheep better.

YOLO algorithms are single-stage object detection algorithms that prioritise speed for accuracy, making them optimal for real-time detection. However, if real-time detection is not a critical requirement, it would be interesting to use two-stage object detection algorithms that could achieve higher accuracy and see how it performs at detecting occluded sheep.

Bibliography

[1] *Beitebruk og seterdrift*, no, May 2021. [Online]. Available: https://www.ssb.no/jord-skog-jakt-og-fiskeri/artikler-og-publikasjoner/beitebruk-og-seterdrift (visited on 18/02/2023).

[2] S.-O. Hvasshovd, *Droner og Sau og litt til !! Anvendelser og Muligheter*, 2017. [Online]. Available: https://www.statsforvalteren.no/contentassets/cbf122460efa4e37a051c17c07fade0d/droner-buskerud-2017.pdf (visited on 19/02/2023).

[3] *Tap av sau på beite*, nb-NO, 2022. [Online]. Available: https://www.dyrebeskyttelsen.no/tap-sau-pa-beite/ (visited on 19/02/2023).

[4] *Gjetargut*. [Online]. Available: https://gjetargut.no/ (visited on 21/02/2023).

[5] *Findmy | GPS Sporing av husdyr på utmarksbeite*. [Online]. Available: https://findmy.no/no/agtech (visited on 21/02/2023).

[6] *Telespor*, nb-NO. [Online]. Available: http://www.telespor.no/ (visited on 21/02/2023).

[7] *Nyhet! Smartbjella 2 - opptil 10år batteri**, nb-NO. [Online]. Available: https://smartbjella.no/ (visited on 22/02/2023).

[8] K. M. Johannessen, 'Towards Improved Sheep Roundup - Using Deep Learning-Based Detection on MultiChannel RGB and Infrared UAV Imagery,' en, Accepted: 2021-09-20T16:04:51Z, M.S. thesis, NTNU, 2020. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2779322 (visited on 15/11/2022).

[9] H. Sørensen Bøckman, 'Locating sheep in the highlands with aerial footage and a lightweight algorithm system.,' eng, Accepted: 2022-03-12T18:19:45Z, M.S. thesis, NTNU, 2021. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2984882 (visited on 14/03/2023).

[10] J. H. Muribø, 'Locating Sheep with YOLOv3,' eng, Accepted: 2019-09-26T14:05:52Z, M.S. thesis, NTNU, 2019. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2619041 (visited on 21/02/2023).

[11] I. Nygård and S. Vittersø, 'Improved Sheep Detection - Modifying YOLOv5 to accurately detect grazing sheep in UAV imagery,' eng, Accepted: 2022-12-22T18:19:32Z, M.S. thesis, NTNU, 2022. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3039298` (visited on 14/03/2023).

[12] J. Kim, S. Kim, C. Ju and H. I. Son, 'Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications,' *IEEE Access*, vol. 7, pp. 105 100–105 115, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2019.2932119`.

[13] B. Østtveit, 'Using synthetic data to improve the detection of sheep in drone images,' eng, Accepted: 2022-10-07T17:31:39Z, M.S. thesis, NTNU, 2022. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3024704` (visited on 15/11/2022).

[14] K. Oksuz, B. C. Cam, S. Kalkan and E. Akbas, *Imbalance Problems in Object Detection: A Review*, en, arXiv:1909.00169 [cs], Mar. 2020. [Online]. Available: `http://arxiv.org/abs/1909.00169` (visited on 05/03/2023).

[15] B. Vanherle, S. Moonen, F. Van Reeth and N. Michiels, *Analysis of Training Object Detection Models with Synthetic Data*, arXiv:2211.16066 [cs], Nov. 2022. DOI: `10.48550/arXiv.2211.16066`. [Online]. Available: `http://arxiv.org/abs/2211.16066` (visited on 01/03/2023).

[16] K. Saleh, S. Szénási and Z. Vámossy, 'Occlusion Handling in Generic Object Detection: A Review,' in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Jan. 2021, pp. 000 477–000 484. DOI: `10.1109/SAMI50585.2021.9378657`. (visited on 03/03/2023).

[17] O. K. Furseth and A. O. Granås, 'Real-time Sheep Detection - Improving Retrieval of Free-ranging Sheep Using Deep Learning-based Detection on Drone Imagery Running on Mobile Devices,' eng, Accepted: 2021-12-15T18:19:55Z, M.S. thesis, NTNU, 2021. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2834578` (visited on 08/11/2022).

[18] H. Stemshaug, 'Impact of Low Resolution IR Images in Drone Based Sheep Detection,' eng, Accepted: 2022-10-18T17:20:52Z, M.S. thesis, NTNU, 2022. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3026821` (visited on 15/11/2022).

[19] T. Vucic and C. Axell, 'Tracking sheep by radio tags and UAV: A field study of Bluetooth round-trip time ranging and multilateration,' eng, Accepted: 2022-10-07T17:31:27Z, M.S. thesis, NTNU, 2022. [Online]. Available: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3024696` (visited on 14/02/2023).

[20] L. Zhang, R. Tanno, M.-C. Xu, C. Jin, J. Jacob, O. Cicarrelli, F. Barkhof and D. Alexander, 'Disentangling Human Error from Ground Truth in Segmentation of Medical Images,' in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 15 750–15 762. [Online]. Available: `https://proceedings.neurips.cc/paper/2020/hash/b5d17ed2b502da15aa727af0d51508d6-Abstract.html` (visited on 22/04/2023).

[21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, 'Domain randomization for transferring deep neural networks from simulation to the real world,' in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Sep. 2017, pp. 23–30. DOI: `10.1109/IROS.2017.8202133`. (visited on 01/04/2023).

[22] P. De Roovere, S. Moonen, N. Michiels and F. Wyffels, *Dataset of Industrial Metal Objects*, arXiv:2208.04052 [cs], Aug. 2022. DOI: `10.48550/arXiv.2208.04052`. [Online]. Available: `http://arxiv.org/abs/2208.04052` (visited on 31/05/2023).

[23] V. Seib, B. Lange and S. Wirtz, *Mixing Real and Synthetic Data to Enhance Neural Network Training – A Review of Current Approaches*, arXiv:2007.08781 [cs], Jul. 2020. DOI: `10.48550/arXiv.2007.08781`. [Online]. Available: `http://arxiv.org/abs/2007.08781` (visited on 03/04/2023).

[24] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, C. Romero, W. Smith, A. Thaman, S. Warren and N. Yadav, *Unity Perception: Generate Synthetic Data for Computer Vision*, arXiv:2107.04259 [cs], Jul. 2021. DOI: `10.48550/arXiv.2107.04259`. [Online]. Available: `http://arxiv.org/abs/2107.04259` (visited on 02/03/2023).

[25] T. Agrawal, *Imbalanced Data in Object Detection Computer Vision Projects*, en-US, Jul. 2022. [Online]. Available: `https://neptune.ai/blog/imbalanced-data-in-object-detection-computer-vision` (visited on 31/03/2023).

[26] X. Wang, A. Shrivastava and A. Gupta, *A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection*, en, arXiv:1704.03414 [cs], Apr. 2017. [Online]. Available: `http://arxiv.org/abs/1704.03414` (visited on 06/04/2023).

[27] A. Kortylewski, Q. Liu, A. Wang, Y. Sun and A. Yuille, *Compositional Convolutional Neural Networks: A Robust and Interpretable Model for Object Recognition under Occlusion*, en, arXiv:2006.15538 [cs], Jun. 2020. [Online]. Available: `http://arxiv.org/abs/2006.15538` (visited on 03/04/2023).

[28] N. K. A. REDAKTØR, *Hvor plagsomt er det for sauen å gå med ei bjelle rundt halsen hele sommeren?* no, Section: naturvitenskap, Jul. 2019. [Online]. Available: `https://forskning.no/dyreverden-husdyr-landbruk/hvor-plagsomt-er-det-for-sauen-a-ga-med-ei-bjelle-rundt-halsen-hele-sommeren/1360767` (visited on 14/03/2023).

[29] *Nofence - Verdens første virtuelle gjerder for beitedyr*, no. [Online]. Available: `https://www.nofence.no` (visited on 01/03/2023).

[30] J. R. E. Johanssen and K. Sørheim, 'Sau - Atferd og velferd hos sau,' no, Norsk senter for økologisk landbruk, Tingvoll, Report, Oct. 2018, ISBN: 9788282020657, pp. 1–6. [Online]. Available: `https://orgprints.org/id/eprint/33947/` (visited on 27/04/2023).

[31] *NIBIO Driftsgranskingane i jord- og skogbruk - Hovudtabellar*. [Online]. Available: `https://driftsgranskingane.nibio.no/drgr/hovudtabellar/?vis=htab&tabell_id=52&aar=2021&lang=BM` (visited on 09/03/2023).

[32] *DJI Mavic 3 Enterprise (Worry-Free Basic Combo)*, no. [Online]. Available: `https://djioslo.no/produkt/enterprise/dji-mavic-3e/` (visited on 28/02/2023).

[33] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, 'Object Detection in 20 Years: A Survey,' *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: `10.1109/JPROC.2023.3238524`.

[34] G. Boesch, *Object Detection in 2023: The Definitive Guide*, en-US, Feb. 2023. [Online]. Available: `https://viso.ai/deep-learning/object-detection/` (visited on 31/05/2023).

[35] R. Szeliski, *Computer Vision: Algorithms and Applications* (Texts in Computer Science), en. Cham: Springer International Publishing, 2022, ISBN: 978-3-030-34372-9. DOI: `10.1007/978-3-030-34372-9`. [Online]. Available: `https://link.springer.com/10.1007/978-3-030-34372-9` (visited on 12/05/2023).

[36] A. Neubeck and L. Van Gool, 'Efficient Non-Maximum Suppression,' in *18th International Conference on Pattern Recognition (ICPR'06)*, ISSN: 1051-4651, vol. 3, Aug. 2006, pp. 850–855. DOI: `10.1109/ICPR.2006.479`.

[37] P. Huilgol, *Precision and Recall | Essential Metrics for Data Analysis (Updated 2023)*, en, Sep. 2020. [Online]. Available: `https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/` (visited on 13/04/2023).

[38] *F-Score*, May 2019. [Online]. Available: `https://deepai.org/machine-learning-glossary-and-terms/f-score` (visited on 19/05/2023).

[39] J. Czakon, *F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?* en-US, Jul. 2022. [Online]. Available: `https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc` (visited on 01/06/2023).

[40] Z. Zhang, 'Artificial Neural Network,' en, in *Multivariate Time Series Analysis in Climate and Environmental Research*, Z. Zhang, Ed., Cham: Springer International Publishing, 2018, pp. 1–35, ISBN: 978-3-319-67340-0. DOI: `10.1007/978-3-319-67340-0_1`. [Online]. Available: `https://doi.org/10.1007/978-3-319-67340-0_1` (visited on 22/11/2022).

[41] B. Liquet, S. Moka and Y. Nazarathy, *4 General Fully Connected Neural Networks | The Mathematical Engineering of Deep Learning*. [Online]. Available: `https://deeplearningmath.org` (visited on 18/04/2023).

[42] A. Krogh, 'What are artificial neural networks?' en, *Nature Biotechnology*, vol. 26, no. 2, pp. 195–197, Feb. 2008, Number: 2 Publisher: Nature Publishing Group, ISSN: 1546-1696. DOI: `10.1038/nbt1386`. [Online]. Available: `https://www.nature.com/articles/nbt1386` (visited on 09/11/2022).

[43] Z. Brodtman, *The Importance and Reasoning behind Activation Functions*, en, Nov. 2021. [Online]. Available: `https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41` (visited on 27/04/2023).

[44] *Neural Networks and Machine Learning : Networks Course blog for INFO 2040/CS 2850/Econ 2040/SOC 2090*, en, sv. [Online]. Available: `https://blogs.cornell.edu/info2040/2015/09/08/neural-networks-and-machine-learning/` (visited on 01/06/2023).

[45] W. Zhiqiang and L. Jun, 'A review of object detection based on convolutional neural network,' in *2017 36th Chinese Control Conference (CCC)*, ISSN: 1934-1768, Jul. 2017, pp. 11 104–11 109. DOI: `10.23919/ChiCC.2017.8029130`.

[46] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, en, Nov. 2022. [Online]. Available: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53` (visited on 21/05/2023).

[47] D. Scherer, A. Müller and S. Behnke, 'Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,' en, in *Artificial Neural Networks – ICANN 2010*, K. Diamantaras, W. Duch and L. S. Iliadis, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2010, pp. 92–101, ISBN: 978-3-642-15825-4. DOI: `10.1007/978-3-642-15825-4_10`.

[48] M. Basavarajaiah, *Which pooling method is better? Maxpooling vs minpooling vs average pooling*, en, Aug. 2019. [Online]. Available: `https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9` (visited on 19/05/2023).

[49] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, arXiv:2207.02696 [cs], Jul. 2022. DOI: `10.48550/arXiv.2207.02696`. [Online]. Available: `http://arxiv.org/abs/2207.02696` (visited on 11/02/2023).

[50] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez and J. García-Gutiérrez, 'On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data,' en, *Remote Sensing*, vol. 13, no. 1, p. 89, Jan. 2021, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs13010089. [Online]. Available: https://www.mdpi.com/2072-4292/13/1/89 (visited on 20/05/2023).

[51] P. Soviany and R. T. Ionescu, *Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction*, arXiv:1803.08707 [cs], Aug. 2018. [Online]. Available: http://arxiv.org/abs/1803.08707 (visited on 20/05/2023).

[52] J. Terven and D. Cordova-Esparza, *A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond*, arXiv:2304.00501 [cs], Apr. 2023. DOI: 10.48550/arXiv.2304.00501. [Online]. Available: http://arxiv.org/abs/2304.00501 (visited on 24/04/2023).

[53] J. Hui, *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*, en, Sep. 2022. [Online]. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088 (visited on 01/06/2023).

[54] Crypto1, *How Does the Gradient Descent Algorithm Work in Machine Learning?* en, Oct. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/ (visited on 21/04/2023).

[55] M. Alloghani, D. Al-Jumeily Obe, J. Mustafina, A. Hussain and A. Aljaaf, 'A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science,' in Jan. 2020, pp. 3–21, ISBN: 978-3-030-22474-5. DOI: 10.1007/978-3-030-22475-2_1.

[56] *Fine Tuning YOLOv7 - Custom Object Detection Training*, en-US, Aug. 2022. [Online]. Available: https://learnopencv.com/fine-tuning-yolov7-on-custom-dataset/ (visited on 23/04/2023).

[57] A. Bronshtein, *Train/Test Split and Cross Validation in Python*, en, Mar. 2020. [Online]. Available: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6 (visited on 12/05/2023).

[58] *Train Test Validation Split: How To & Best Practices [2023]*, en. [Online]. Available: https://www.v7labs.com/blog/train-validation-test-set,%20https://www.v7labs.com/blog/train-validation-test-set (visited on 13/05/2023).

[59] *COCO - Common Objects in Context*. [Online]. Available: https://cocodataset.org/#home (visited on 02/05/2023).

[60] *The PASCAL Visual Object Classes Homepage*. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/ (visited on 07/05/2023).

[61] *Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine*, en. [Online]. Available: `https://unity.com` (visited on 25/04/2023).

[62] *Denis Pahunov / MapMagic World Generator · GitLab*, en. [Online]. Available: `https://gitlab.com/denispahunov/mapmagic` (visited on 27/04/2023).

[63] *HDRP Oak Tree | 3D | Unity Asset Store*, en. [Online]. Available: `https://assetstore.unity.com/packages/3d/hdrp-oak-tree-214007` (visited on 11/05/2023).

[64] *HDRP Pine Tree | 3D Trees | Unity Asset Store*, en. [Online]. Available: `https://assetstore.unity.com/packages/3d/vegetation/trees/hdrp-pine-tree-214095` (visited on 11/05/2023).

[65] *SHEEP | Characters | Unity Asset Store*, en. [Online]. Available: `https://assetstore.unity.com/packages/3d/characters/animals/mammals/sheep-5012` (visited on 11/05/2023).

[66] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Generative Adversarial Networks*, arXiv:1406.2661 [cs, stat], Jun. 2014. DOI: `10.48550/arXiv.1406.2661`. [Online]. Available: `http://arxiv.org/abs/1406.2661` (visited on 19/05/2023).

[67] J. Brownlee, *A Gentle Introduction to Generative Adversarial Networks (GANs)*, en-US, Jun. 2019. [Online]. Available: `https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/` (visited on 20/05/2023).

[68] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, *Analyzing and Improving the Image Quality of StyleGAN*, en, arXiv:1912.04958 [cs, eess, stat], Mar. 2020. [Online]. Available: `http://arxiv.org/abs/1912.04958` (visited on 19/05/2023).

[69] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat, R. Laganiere and J. Rebut, *How much real data do we actually need: Analyzing object detection performance using synthetic and real data*, arXiv:1907.07061 [cs], Jul. 2019. DOI: `10.48550/arXiv.1907.07061`. [Online]. Available: `http://arxiv.org/abs/1907.07061` (visited on 28/03/2023).

Appendix A

Dataset and Code Repository

Project for generating synthetic images in Unity Perception:

- `https://github.com/Borern/master-unity`

Synthetic and real image sheep dataset:

- `https://www.kaggle.com/datasets/sindrelangaard/uav-sheep-dataset`